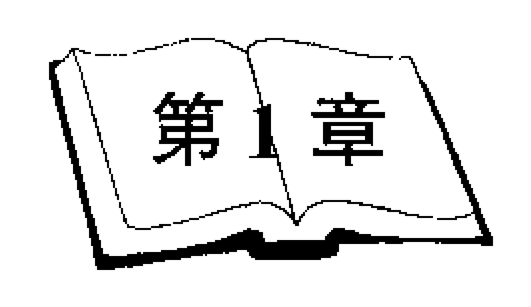
目　录



概　论

1.1　数值计算方法

1.1.1　什么是数值计算方法

数值计算方法是在解决科学研究和工程实践中遇到的复杂问题的长期过程中形成的一门学科。

一般而言，在科学研究和工程实践中遇到的各种复杂的实际问题，通常可以用物理模型和数学模型来描述。通过对数学模型(物理模型一般转化成数学模型来研究)进行理论分析和求解，最终得到问题的解。然而，并不是所有的数学模型即数学问题都能够采用数学分析的方法来进行求解。

例如：若已知两个变量 x 和 y 之间的一组离散数据$(x_i,y_i)(i=0,1,\cdots,n)$，要通过这组数据找到变量 x 和 y 之间的函数关系，即数学表达式 $y=f(x)$，则该问题直接用数学分析方法来解决就比较困难。

对于常微分方程的初值问题，例如，一阶常微分方程的初值问题

$$\begin{cases} y'(x)=f(x,y), x\in[a,b] \\ y(x_0)=y_0 \end{cases}$$

用解析法一般只能求得某些类型微分方程的解 $y=y(x)$ 而不能得到所有微分方程的解。

对于定积分问题，若函数 $f(x)$ 在区间$[a,b]$上连续且其原函数为 $F(x)$，则有：

$$\int_a^b f(x)\mathrm{d}x=F(b)-F(a)$$

但是，如果当被积函数 $f(x)$ 不能找到用初等函数有限形式表示的原函数时，例如：

$$f(x)=\frac{\sin x}{x},\sin x^2,\cos x^2,\frac{1}{\ln x},\mathrm{e}^{-x^2},\sqrt{1+x^3},\cdots$$

或者当 $y=f(x)$ 之间的函数关系是以离散的数据$(x_i,y_i)(i=0,1,\cdots,n)$来表示时，则定积分的计算是比较困难的。

还有许多实际问题在求解其数学问题的解时，虽然能够证明解的存在性与惟一性，但要给出解析解则是很困难的。可见，数学分析方法存在着一定的局限性。

怎样解决这类问题呢？下面仅以一阶常微分方程的初值问题为例，说明如何用数值计算方法来解决这类问题。

由一阶常微分方程可得到 x_n 点处的导数 $y'(x_n)$，即：

$$y'(x_n)=f(x_n,y_n)$$

再利用导数和差商的近似关系

$$y'(x)=\lim_{\Delta x\to 0}\frac{\Delta y}{\Delta x}=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}\cong\frac{y(x_{n+1})-y(x_n)}{x_{n+1}-x_n}$$

得到差分方程为：

$$\frac{y(x_{n+1})-y(x_n)}{x_{n+1}-x_n}=\frac{y(x_{n+1})-y(x_n)}{h}=f(x_n,y_n)$$

从而得到一阶常微分方程解 $y=y(x)$ 的离散值的递推计算公式为：

$$y(x_{n+1})=y(x_n)+hf(x_n,y(x_n))$$

由该递推公式，可以计算出一阶常微分方程在任意给定点 $x_k(k=1,2,\cdots)$ 的数值解 $y(x_k)$。

由此可见，一个连续的数学问题最终可以转化为一个离散的数学问题来解决，即连续的数学问题最终可按照一定的规则进行一系列的算术运算和逻辑运算而得到计算结果。这就是所谓的数值计算或科学计算。因此，数值计算方法就是研究如何把数学问题归结为数值计算的问题(即如何构造数值计算问题的算法)，且当实际问题的精确解无法得到时，如何才能得到足够准确的数值解，以及解的特性和各种计算方法的优缺点及其适用范围。或者说，数值计算方法就是研究各种数学问题数值解法的理论和方法，包括计算方法的构造与求解过程的理论分析和算法结构。因此相对于数学分析而言，数值计算方法又被称为数值分析。数值计算方法是寻求数学问题近似解的方法、过程及其理论的一个重要数学分支。

应该注意，在本书中数值计算方法和计算方法在概念上是有所区别的。数值计算方法是指研究各种数学问题数值解法理论和方法的总称，是一门学科；而计算方法则是指具体解决某一数学问题的数值计算的方法和算法，两者不应混淆。

数值计算方法在科学研究和生产实践中得到了广泛应用。最近半个世纪以来，科学研究的实践使人们越来越清楚地认识到，当代科学研究方法论应该由理论、实验和科学计算三大环节所组成，科学计算已成为科学研究方法的重要组成部分。

例如，在科学计算中遇到的复杂问题可能是：

(1) 采用数学分析方法太难的设计计算；

(2) 需要反复多次的计算；

(3) 采用人工计算太繁的设计计算；

(4) 大量的数据处理。

这些计算问题仅仅依靠人工计算的方法是不现实的，甚至是不可能的，必须采用计算机来进行计算。由于计算机只能依据给定的指令(对一定数位的数)完成加、减、乘、除四则算术运算和一些逻辑运算，因此计算机特别适合用于数值计算。随着计算机性能的不断提高，计算机运算速度越来越快，存储数据的容量越来越大，人们往往更喜欢使用计算机来帮助解决实际问题。一个实际问题采用计算机来求解的具体步骤如图1.1所示。



图1.1　应用计算机解决实际问题的过程

因此，学好数值计算方法这门课程十分重要。

1.1.2　数值计算方法的特点

数值计算方法具有以下几个特点：

1）具有理论和技术的二重性

数值分析是相对于数学分析(或理论分析)的一种数学方法，它不同于纯数学那样只研究数学本身的问题，而是着重研究数学问题的数值解法及其相关理论(数值解的收敛性、稳定性和误差分析等)，但它仍然是以一定的数学理论为基础进行的。有些计算方法虽然在理论上并不完善，但通过实际计算、对比分析等手段，被证明是行之有效的经常会采用；而有些计算方法虽然在理论上较为完善，但通过实际计算是不可行的却常常要放弃。因此，数值计算方法既具有数学分析的理论性、抽象性与严谨性，又具有实际应用的技术性和可行性的特点。

2）计算过程是面向计算机的

计算过程是面向计算机还是面向人工计算，存在着较大区别。随着计算机的迅速发展，数值计算方法应该面向计算机，应该根据数字计算机的特点(计算机只能进行加、减、乘、除四则算术运算和一些逻辑运算)，提出实际可行的、有效的和计算复杂性(计算精度、时间复杂性和空间复杂性等)适当的计算方法。

3）计算方法可以用数值实验证实

某些计算方法的优劣程度、可行性和有效性等完全可以通过数值实验得到证明和验证。数值实验和数学理论分析一样，都是数值计算方法的重要研究手段。

1.1.3 数值计算方法的常用算法

使用计算机进行数值计算，把对数学问题的求解运算转化为有限数位数的有限四则运算和逻辑运算，并对运算顺序有完整而准确的描述和规定，就是数值计算方法的算法。通常有离散化算法、递推算法、逼近算法和构造性算法等。

1. 离散化方法

离散化方法就是通过极限方法、逼近方法等基本思想和方法，把连续性的数学问题转化成离散的数学问题来处理。

例1.1 定积分的数值计算方法：

$$I=\int_a^b f(x)\mathrm{d}x$$

解 由于计算机无法计算这个连续性的数学问题，故可将此问题转化为离散的数学问题来处理，即：

(1) 将区间$[a,b]$分成n等分，即：

$a=x_0<x_1<\cdots<x_n=b$，其中$x_i=x_0+ih\ (i=1,2,\cdots,n)$

$$h=\frac{b-a}{n}$$

(2) 将每个小区间$[x_{i-1},x_i]\ (i=1,2,\cdots,n)$所对应的小曲边形的面积$S_i$用小梯形面积近似代替(如图1.2所示)，即：

$$S_i\approx\frac{h}{2}[f(x_{i-1})+f(x_i)]$$

图1.2 定积分近似计算

(3) 求和：

$$S=\sum_{i=1}^{n}S_i\approx\frac{h}{2}\sum_{i=1}^{n}[f(x_{i-1})+f(x_i)]=\frac{h}{2}\left[f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right]$$

最后得到定积分的数值计算公式为：

$$\int_a^b f(x)\mathrm{d}x \approx \frac{h}{2}\left[f(a)+2\sum_{i=1}^{n-1}f(x_i)+f(b)\right]$$

2. 递推方法

递推方法就是构造关于离散变量之间的计算公式，并可以由某个已知的离散变量值(即初始值)逐步推导出所有的离散变量值。

例1.2 求多项式$p_n(x)=a_0+a_1x+a_2x^2+\cdots+a_nx^n$在给定点$x$处的值(当各项系数和$n$已知时)。

解 取中间变量t和u，设$t_0=1, u_0=a_0$，则有：

$$t_1=xt_0=x^1 \quad u_1=u_0+a_1t_1=a_0+a_1x$$

$$t_2=xt_1=x^2 \quad u_2=u_1+a_2t_2=a_0+a_1x+a_2x^2$$

$$t_3=xt_2=x^3 \quad u_3=u_2+a_3t_3=a_0+a_1x+a_2x^2+a_3x^3$$

最后可以得到：

$$\begin{cases} u_k=u_{k-1}+a_kt_k \\ t_k=xt_{k-1} \end{cases}, \quad k=1,2,3,\cdots,n$$

其中，$t_0=1, u_0=a_0$。显然，通过上述递推公式，可由u_{k-1}求得u_k，并且有$p_n(x)=u_n$。

假设在递推化方法中，x_{n+1}由x_n递推计算出来。如果当n趋向∞时，x_{n+1}趋向某个常数C，则称之为迭代法。例如，设有$x_n=\frac{1}{2}\left(x_{n-1}+\frac{a}{x_{n-1}}\right)(a>0; x_0>0; n=1,2,3,\cdots)$，当$n\to\infty$时，$x_n\to\sqrt{a}$，$x_n$的递推计算公式即为迭代公式，它可以用来计算$\sqrt{a}$的近似值。

3. 逼近方法

逼近是指用简单函数$p(x)$的值近似代替函数$f(x)$的值，$f(x)$称为被逼近函数，$p(x)$称为逼近函数，两者的差$e(x)=f(x)-p(x)$称为逼近误差或余项。

所谓简单函数是指可以用四则运算进行计算的函数，如多项式、有理函数等。插值和拟合是常见的逼近方法。

例1.3 计算无理数e的近似值。

解 对于函数$f(x)=e^x$，在$x=0$处用Taylor公式展开，得

$$e^x=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}+\cdots$$

取$x=1$，得

$$e=1+1+\frac{1}{2!}+\cdots+\frac{1}{n!}+\cdots$$

这是一个无限求和过程，无法进行实际计算。根据逼近方法，可得到无理数e的计算公式

$$e\approx 1+1+\frac{1}{2!}+\cdots+\frac{1}{n!}$$

再利用Taylor公式的余项进行误差估计，即：

$$R_n(x)=e^x-\left(1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}\right)=\frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}, \xi\in(0,1)$$

所以，无理数e计算公式的误差为：

$$|R_n(1)|\leqslant\frac{e}{(n+1)!}<\frac{3}{(n+1)!}$$

它与n有关。使用逼近法时，一定要进行误差分析。本例中的逼近法常常称为截断法。

4. 构造方法

某些计算方法在证明一个数学问题的解的存在性时，可使用所谓构造式的方法。构造方法不仅能够证明解的存在，而且可以同时给出解的具体表达式(或近似计算公式)。

例1.4 证明一元二次方程$ax^2+bx+c=0,a\neq0$，当$b^2-4ac>0$时有两相异实根。

证 $ax^2+bx+c=a\left(x+\frac{b}{2a}\right)^2+\frac{4ac-b^2}{4a}=0$， $a\left(x+\frac{b}{2a}\right)^2=\frac{b^2-4ac}{4a}$

所以，当$b^2-4ac>0$时有两相异实根：

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a},\quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

应当注意，在实际解决一个具体问题时，往往会同时用到数值计算方法的几种算法。

1.1.4 数值计算方法的主要内容

能够用数值计算方法来解决的数学问题十分广泛。本书主要讨论以下主要内容：非线性方程的求根、插值与曲线拟合、数值积分、常微分方程的数值解法、线性方程组的数值解法等。

学习数值计算方法这门课程，应该针对要解决数学问题的类型，掌握其计算方法的基本原理、基本思想和方法技巧。在具体应用数值计算方法解决实际问题时，还要重视数值精度、收敛性和稳定性、计算量等问题，因为它是评价一个计算方法优劣的重要指标。最后通过例题学习和大量练习，学会灵活使用各种数值方法来解决实际的计算问题。

MATLAB程序语言是科学计算的计算机高级编程语言，本书第7章系统而简明地介绍了MATLAB语言编程的基本内容，并且全国使用MATLAB语言编制有关数值计算方法的具体算法程序，附录A给出了部分数值计算方法的程序。

1.2 误差和有效数字

1.2.1 误差和误差限

在数值计算中使用的数通常有两种类型：一种是能够准确反映客观事物数量关系的精确数，如一个班级有30名学生，其中1/2为男生，1/2为女生，男生拥有的电脑数为女生的3倍，等等，这些数和实际值之间没有任何差异；另一种是近似反映客观事物数量关系的近似数，如一本书重0.15kg，一个物体长20cm等，如果改变观测方法、提高测量精度，这些数据虽然会更准确些，但仍然会和实际值之间存在着一定差异，这种差异就是误差。在数值计算中用到的数据大部分情形都是近似数，因此控制计算误差、提高计算精度是十分重要的问题。下面将讨论有关误差的问题。

1. 绝对误差和绝对误差限

设数x^*为精确数(精确值或真值)，数x为其近似数(近似值)，则

$$e(x)=x^*-x \tag{1.1}$$

称为近似值x的绝对误差。一般情况下,人们无法准确地知道精确数x^*的大小,但根据具体的测量和计算情况,如果总有

$$|e(x)|=|x^*-x|\leqslant\varepsilon \tag{1.2}$$

成立,则称ε为近似值x的绝对误差限。显然有

$$x-\varepsilon\leqslant x^*\leqslant x+\varepsilon \tag{1.3}$$

这表明精确数介于某两个数之间。因此在工程上,精确值x^*常常可表示成另一种形式:

$$x^*=x\pm\varepsilon \tag{1.4}$$

例 1.5 用一把最小刻度为1mm的尺子,测量桌子的长度,读出来的值是$x=1235$mm,这是桌子长度x^*的近似值,此近似值的绝对误差为:

$$|x^*-x|=|x^*-1235|\text{mm}\leqslant\frac{1}{2}\text{mm}$$

所以

$$1234.5\text{mm}\leqslant x^*\leqslant 1235.5\text{mm}$$

这表明桌子长度的真值x^*在[1234.5,1235.5]之间,即

$$x^*=(1235\pm 0.5)\text{mm}$$

这里,绝对误差限ε为近似值x末位的半个单位。

例 1.6 用游标卡尺测量某个物体的长度x^*,其读数是$x=12.56$mm。由于游标卡尺的最小刻度为0.02mm,故物体的长度x^*应记为:

$$x^*=(12.56\pm 0.02)\ \text{mm}$$

这里,绝对误差限ε为近似值x末位的2个单位。

绝对误差不足以刻画近似数的精确程度,如用游标卡尺测量一个物体的尺寸,其误差为0.1mm是不允许的,而测量一个人的身高,误差为1mm则是允许的。因此,要决定一个近似值的精确程度,除了绝对误差以外,还必须考虑此数本身的大小,即有相对误差的概念。

2. 相对误差与相对误差限

定义近似数x的相对误差为:

$$e_r(x)=(x^*-x)/x^*$$

由于精确数x^*在一般情形下无法得知,而$x\approx x^*$,故相对误差常用下列定义:

$$e_r(x)=(x^*-x)/x \tag{1.5}$$

相应地,相对误差限ε_r定义为:

$$|e_r(x)|=\left|\frac{x^*-x}{x}\right|\leqslant\varepsilon_r \tag{1.6}$$

1.2.2 有效数字

数学上常用"四舍五入"的法则将一个位数很多的数表示成一定位数的数。如果一个近似数的误差限是它某一位的半个单位,则称它准确到这一位(即该位数字是准确的、有效的和可靠的)。并且,从该位起直到前面第一个非零数字为止的所有数字都称为有效数字,即有:

定义 1.1 设x^*是一个精确数,x是它的近似数,若

$$|x^*-x|<1/2\times 10^{-n} \tag{1.7}$$

就是说,用x近似表示x^*时准确到小数点之后第n位(精确度),并把从该位起到x的第一位非零数字之间的一切数字都叫做有效数字,并把有效数字的位数叫做有效数位。

例如，若将数x用规格化形式表示，则有：

$$x=\pm 0.a_1a_2a_3\cdots a_{s-1}a_s\cdots a_k\times 10^m \quad (a_1\neq 0) \tag{1.8}$$

其中，s为有效数字的位数，m为阶数。m的确定方法如下：

当$|x|\geqslant 1$时，则m=整数部分的位数；

当$0.1\leqslant |x|<1$时，则$m=0$；

当$|x|<0.1$时，则$m=-$（小数点后零的个数）。

若

$$|x-x^*|<1/2\times 10^{-n}=1/2\times 10^{-(s-m)}=1/2\times 10^{m-s},\quad 1\leqslant s\leqslant k \tag{1.9}$$

则x的有效数位为：

$$s=n+m \tag{1.10}$$

应用公式(1.9)和(1.10)可以判断一个数的某一位数字是否为有效数字，从而可确定该数的有效数位。

例1.7 确定数0.08698，0.8698，869.8和8698的有效数位。

解 对于数0.08698，$n=5$，$m=-1$，$s=n+m=4$；

对于数0.8698，$n=4$，$m=0$，$s=n+m=4$；

对于数869.8，$n=1$，$m=3$，$s=n+m=4$；

对于数8698，$n=0$，$m=4$，$s=n+m=4$。

一个数精确到小数点后n位，并不表明它一定有n位有效数字。应注意有效数位s的计算。

例1.8 对于圆周率$\pi=3.1415926\cdots$，用四舍五入取小数点后4位时，近似值为3.1416，此时

$$|\pi-3.1416|\leqslant\frac{1}{2}\times 10^{-4}$$

即有$m=1$，$n=4$，所以有效数位$s=n+m=5$，绝对误差限为$1/2\times 10^{-4}$。

如果取$\pi=3.14$，则$m=1$，$n=2$，有效数位$s=n+m=3$，其绝对误差限为$1/2\times 10^{-2}$。

例1.9 设$x^*=8.000033$，若取$x=8.0000$，则

$$|x-x^*|=0.000033=0.33\times 10^{-4}<1/2\times 10^{-4}=1/2\times 10^{1-5}$$

于是有$m=1$，$s=5$，$n=s-m=5-1=4$。所以，8.0000的有效数字为5位，它精确到小数点后4位。

例1.10 设$x^*=30.4500364\cdots$，若取$x=30.45004$，则

$$|x-x^*|=0.0000036=0.36\times 10^{-5}<1/2\times 10^{-5}=1/2\times 10^{2-7}$$

所以，30.45004的有效数字为7位，它精确到小数点后5位。

1.2.3 相对误差和有效数位之间的关系

根据有效数字和相对误差的概念可以得出两者之间的关系。

定理1.1 若近似数$x=\pm 0.a_1a_2a_3\cdots a_{s-1}a_s\times 10^m(a_1\neq 0)$具有$s$位有效数字，则其相对误差为：

$$e_r(x)\leqslant\varepsilon_r=\frac{1}{2a_1}\times 10^{-s+1} \tag{1.11}$$

证明 由于x有s位有效数字，故可设

$$|x^*-x|\leqslant\frac{1}{2}\times 10^{m-s}$$

又　　$|x| = 0.a_1a_2a_3\cdots a_{s-1}a_s \times 10^m = a_1.a_2a_3\cdots a_{s-1}a_s \times 10^{m-1}$

故　　$|x| \geqslant a_1 \times 10^{m-1}$

所以　　$$\frac{|x^*-x|}{|x|} \leqslant \frac{1/2\times 10^{m-n}}{a_1\times 10^{m-1}} = \frac{1}{2a_1}\times 10^{-s+1}$$

例 1.11　讨论 π 在四舍五入时取不同位数有效数字的相对误差。

解　$\pi=3.14$　　$s=3$　　$\varepsilon_r=\frac{1}{2\times 3}\times 10^{-3+1}=0.17\%$

$\pi=3.142$　　$s=4$　　$\varepsilon_r=\frac{1}{2\times 3}\times 10^{-4+1}=0.017\%$

$\pi=3.14159$　　$s=6$　　$\varepsilon_r=\frac{1}{2\times 3}\times 10^{-6+1}=0.00017\%$

可见,有效数字位数越多,相对误差就越小。

例 1.12　求 $\sqrt{6}$ 的近似值,使其相对误差不超过 $1/2\times 10^{-3}$。

解　因为 $\sqrt{6}=2.4494\cdots$,取 $a_1=2$,设 $\sqrt{6}$ 取 s 位有效数字,则

$$\frac{1}{2a_1}\times 10^{-s+1}=\frac{1}{2\times 2}\times 10^{-s+1}=\frac{1}{2}\times 10^{-3}$$

$$s=3\cdots,\text{故 } s\geqslant 4,\ \sqrt{6}\approx 2.449。$$

定理 1.2　若近似数 $x=\pm 0.a_1a_2a_3\cdots a_{s-1}a_s\times 10^m (a_1\neq 0)$ 的相对误差

$$e_r(x)\leqslant \frac{1}{2(a_1+1)}\times 10^{-s+1} \tag{1.12}$$

则该近似数有 s 位有效数字。

证明　由于 $|x|=a_1.a_2a_3\cdots a_{s-1}a_s\times 10^{m-1}$,故有:

$$|x|\leqslant (a_1+1)\times 10^{m-1}$$

$$|x^*-x|=\frac{|x^*-x|}{|x|}\times |x|\leqslant \frac{1}{2(a_1+1)}\times 10^{-s+1}\times (a_1+1)\times 10^{m-1}=\frac{1}{2}\times 10^{m-s}$$

因此,x 有 s 位有效数字。

从误差和有效数字的定义以及上述两个定理可以看出:绝对误差与小数点后的位数(数的精确度)有关;相对误差与有效数字的位数有关。

1.2.4　近似数算术运算的误差和有效数位

1. 近似数算术运算的误差

给定函数 $u=f(x,y)$,设 x,y 分别为 x^*,y^* 的近似值,则绝对误差为:

$$e(u)=f(x^*,y^*)-f(x,y)\approx \frac{\partial f(x,y)}{\partial x}(x^*-x)+\frac{\partial f(x,y)}{\partial y}(y^*-y)$$

即
$$e(u)=\frac{\partial f(x,y)}{\partial x}e(x)+\frac{\partial f(x,y)}{\partial y}e(y) \tag{1.13}$$

而相对误差为 $e_r(u)=\frac{e(u)}{u}\approx \frac{\partial f(x,y)}{\partial x}\frac{e(x)}{u}+\frac{\partial f(x,y)}{\partial y}\frac{e(y)}{u}$,即

$$e_r(u)=\frac{\partial f(x,y)}{\partial x}\frac{x}{u}e_r(x)+\frac{\partial f(x,y)}{\partial y}\frac{y}{u}e_r(y) \tag{1.14}$$

由 1.13 和 1.14 两式可得到两个近似数进行算术运算的计算公式为:

$$e(x+y)=e(x)+e(y),\quad e_r(x+y)=\frac{x}{x+y}e_r(x)+\frac{y}{x+y}e_r(y) \tag{1.15}$$

$$e(x-y)=e(x)-e(y),\quad e_r(x-y)=\frac{x}{x-y}e_r(x)+\frac{y}{x-y}e_r(y) \tag{1.16}$$

$$e(x\cdot y)\approx y\cdot e(x)+x\cdot e(y),\quad e_r(x\cdot y)\approx e_r(x)+e_r(y) \tag{1.17}$$

$$e\left(\frac{x}{y}\right)\approx\frac{y\cdot e(x)-x\cdot e(y)}{y^2},\quad e_r\left(\frac{x}{y}\right)\approx e_r(x)-e_r(y),y\neq 0 \tag{1.18}$$

以上给出了两个近似数进行算术运算的误差计算公式.对于多个近似数的算术运算的误差计算公式可以类似导出。

2. 近似数算术运算的有效数位

1）加减运算

近似数进行加减运算时，把其中小数位数较多的数四舍五人，使其比小数位数最少的数多一位小数，计算结果保留的小数位数与原近似数中小数位数最少者相同。例如：

$$x=2.718,y=3.40823$$

$$x+y\approx 2.718+3.4082\approx 6.126$$

$$x-y\approx 2.718-3.4082\approx -0.690$$

2）乘除运算

近似数进行乘除运算时，各因子保留的位数应比有效数字位数最少者的位数多一位，所得结果的有效数字位数与原近似数中有效数字位数最少者的位数最多少一位。例如：

$$x=2.501,y=1.2$$

$$x\times y=2.50\times 1.2\approx 3.0$$

$$x\div y=2.50\div 1.2\approx 2.1$$

3）乘方与开方运算

近似数在进行乘方与开方运算时，原来近似数有几位有效数字，计算结果仍然保留几位有效数字。例如：

$$x=1.2$$

$$x^2\approx 1.4\quad \sqrt{x}\approx 1.1$$

4）对数运算

近似数在进行对数运算时，所取对数的位数与其真数的有效数字位数相同。例如：

$$\lg 2.718\approx 0.4343,\quad \lg 0.0618\approx\bar{2}.791\approx -1.21$$

在实际计算过程中，中间计算结果应比上述各法则规定的位数多一位，在进行最后一次计算时这一位要进行四舍五人。

1.3 计算方法的稳定性

在实际数值计算过程中，由于不可避免地存在和不断产生各种误差，因此计算结果不是绝对精确的。如果误差使得计算结果和实际情况有较大差别或者出现错误的结果，则数值计算便失去了价值和意义。因此，分析数值计算过程中误差的来源和传递规律，设法控制和减小误差，是非常必要和重要的。

1.3.1 误差的来源

数值计算结果中的误差通常根据来源可分为固有误差与计算误差两大类，即：

$$\text{误差来源}\begin{cases}\text{固有误差}\begin{cases}\text{模型误差}\\\text{观测误差}\end{cases}\\\text{计算误差}\begin{cases}\text{截断误差}\\\text{舍入误差}\end{cases}\end{cases}$$

1. 模型误差

用数学模型描述实际问题时,往往抓住主要因素,忽略一些次要因素,即将实际问题理想化,进行数学概括。这种对实际问题进行近似的数学描述时所产生的误差,称为模型误差。

例 1.13 一个物体的重量G与其质量m和重力加速度g的关系为$G=mg$,在国际单位制中,$g\approx 9.81$,由此建立的重量数学模型是:

$$G=9.81m$$

由于g与海拔高度、温度、磁场、地貌结构等诸多因素有关,故会导致在不同的时间和不同的测试地点g的大小不同。因此,上述模型是一个近似模型,它存在的误差就是模型误差。

2. 观测误差

数值问题的原始数据,一般由观测或实验获得,它们和其真值之间存在的误差,称为观测误差。在上例1.13中,假设测量出物体的质量为$m=5.1\text{kg}$,它和质量真值之间存在的误差,就是观测误差。

3. 截断误差

理论上的精确值往往要用无限的过程才能求出,但实际的计算过程只能截取无穷项中的有限项来进行。这种由截断得到的误差,称为截断误差。

例 1.14 根据 Taylor 定理,指数函数e^x可以展开为下列幂级数形式:

$$e^x=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}+\cdots$$

但在实际计算过程中只取有限项之和,即:

$$S_n(x)=1+x+\frac{x^2}{2!}+\cdots+\frac{x^n}{n!}$$

若用$S_n(x)$来近似地表示e^x之值,则其截断误差为:

$$e^x-S_n(x)=\frac{x^{n+1}}{(n+1)!}e^{\theta x},\quad 0<\theta<1$$

另外,用计算机解决科学计算问题时,往往把一个连续变量离散化,如例1.1,这种由离散得到的误差,称为离散误差。

通常,截断误差和离散误差统称为方法误差。

4. 舍入误差

设s是r进制数,p是r进制正负整数或零,则形如

$$x=s\times r^p \tag{1.19}$$

并满足

$$-1<s<1 \tag{1.20}$$

的数x,称为r进制浮点数,且s和p分别称为浮点数的尾数和阶数。

任何一种计算机只能用有限的位数来表示浮点数的尾数和阶数。设

$$-m \leqslant p \leqslant M \tag{1.21}$$

其中m,M为正整数,它们主要由计算机用多少位数来表示阶数而决定。如果尾数的小数尾数为t(t一般比m,M的位数大若干倍),则计算机的数系由一切阶数满足$-1<s<1$的t位r进制浮点数的集合F组成。可见,在计算机数系中,数的个数有限,数系中的每一个数都是有理数,且阶数相等的数以相等的距离分布在数轴的某一段上。

由于计算机数系是有限集,不仅无理数e,π等不属于计算机数系,而且一些有理数(如1/3)也不属于计算机数系,因此常常用计算机数系中和它们接近的数来表示它们。同时,在利用计算机进行计算时,由于字长限制,参与运算的数的长度也是有限的,而由此产生的误差,称为舍入误差。

例1.15 浮点数F的集合也可以用以下4个参数来描述:

$$\{F\}\equiv\{\beta,t,L,U\}$$

其中,β为基数,t是精度参数,整数L与U是阶码E(范围)的下限和上限$[L,U]$。

这样,F中的每一个浮点数x的值可表示为:

$$x=\pm\left(\frac{d_1}{\beta}+\frac{d_2}{\beta^2}+\cdots+\frac{d_t}{\beta^t}\right)\cdot\beta^E$$

式中的整数$d_1\cdots,d_t$满足$0\leqslant d_i\leqslant\beta-1,(i=1,\cdots,t)$,同时有$L\leqslant E\leqslant U$($E$是整数)。

如果对F中每个非零的x,有$d_1\neq0$,则称浮点数系F为规格化浮点数系。括号中的部分$f=(d_1/\beta+d_2/\beta^2+\cdots+d_t/\beta^t)$称为尾数。我们知道,一个实数常用[整数+小数点+尾数]的形式表示,它们在计算机中对应的浮点数$[\beta^t\cdot f]$则常用某种整数表示方式(例如以原码、反码或补码的形式)存储。

例1.16 求十进制数系与计算机采用的二进制数系之间的差别。

解 以在许多算法中常被选作步长的十进制0.1为例。在$\beta=2$或者为2的幂的浮点数系中,10个0.1的步长并不刚好等于一个1.0的步长。事实上,当把1/10转换成为以1/2为底的幂的有限项展开式时,有:

$$\frac{1}{10}=\frac{0}{2^1}+\frac{0}{2^2}+\frac{0}{2^3}+\frac{1}{2^4}+\frac{1}{2^5}+\frac{0}{2^6}+\frac{0}{2^7}+\cdots$$

用下标表示数基,如果$\beta=2$,则有:

$$(0.1)=(0.000110011001100\cdots)_2$$

如果$\beta=8$,则 $$(0.1)=(0.063146314\cdots)_8$$

由于字长限制,等号右边的值只能取7位。很明显,当把10个这样的数相加时,其结果并不正好是1.0。这就是舍入误差造成的。所以,在计算机上进行的浮点运算(四则运算)只能是近似计算。

观测误差和数据的舍入误差虽然来源不同,但对计算结果的影响完全一样;数学模型的误差,往往不是计算工作者所能独立解决的。因此,在数值计算方法这门课程中所涉及到的误差,一般是指舍入误差(包括初始数据误差)和截断误差,并主要讨论它们在计算过程中的传递和对计算结果的影响,以及研究控制它们的影响以保证最终计算结果有足够的精度。

1.3.2 计算方法的稳定性

计算方法的稳定性是指数值计算是否稳定的问题。在数值计算过程中,数值解是逐步计算

出来的。由于计算机的字长有限，每一步计算都有误差存在，且前一步的舍入误差必然要影响下一步的近似解。如果运算序列的舍入误差不增长，误差的积累或传递对计算结果的影响是可控的，则该算法是数值稳定的，否则是数值不稳定的。

例 1.17 计算积分：

$$E_n=\int_0^1 x^n e^{x-1}\mathrm{d}x \quad (n=1,2,\cdots)$$

解 用分部积分法，可得到：

$$\int_0^1 x^n e^{x-1}\mathrm{d}x=x^n e^{x-1}\Big|_0^1-\int_0^1 nx^{n-1}e^{x-1}\mathrm{d}x$$

或
$$E_n=1-n\cdot E_{n-1} \quad (E_1=1/e;n=2,\cdots) \tag{A}$$

用递归公式(A)进行近似计算开头的 9 个值为：

$E_1\approx 0.367879$ $E_4\approx 0.170904$ $E_7\approx 0.110160$

$E_2\approx 0.264242$ $E_5\approx 0.145480$ $E_8\approx 0.118720$

$E_3\approx 0.207274$ $E_6\approx 0.127120$ $E_9\approx -0.068480$

在这些计算结果中居然出现了 E_9 是负值，而据题意，在整个积分区间(0,1)是不会出现负值的。问题发生在哪里呢?

由于(A)式是递归公式，前面计算中惟一的舍入误差在 E_1 处，它只取了 6 位有效数字，以后的计算误差都是由它引起的。E_1 的舍入误差($\approx 4.412\times 10^{-7}$)经(A)式计算$E_2$，使$E_2$ 的舍入误差比E_1 的误差放大了(−2)倍。如此类推，E_9 的误差大约是$(-9!)\times 4.412\times 10^{-7}\cong 0.1601$。计算误差扩大如此迅速，就是因为这里采用的计算方法不适合的缘故。这种情况就是所谓算法不稳定。

但是，若把(A)式改写成可以使误差逐渐减少的下列递归关系：

$$E_{n-1}=\frac{1-E_n}{n} \quad (n=\cdots,3,2) \tag{B}$$

则就可以实现稳定的计算。因为(B)式会使计算的每一步都比前一步的计算舍入误差减少$1/n$倍。在这里，计算的关键是确定计算的起始值E_n。为了确定起始值，可以做如下比较估计：

$$E_n=\int_0^1 x^n e^{x-1}\mathrm{d}x\leqslant\int_0^1 x^n\mathrm{d}x=\frac{x^{n+1}}{n+1}\Big|_0^1=\frac{1}{n+1}$$

于是，当$n\to\infty$时，$E_n\to 0$。因此，可以取E_{20}的近似值为0，则起始误差最大也只有1/21。此误差在求E_{19}时要乘以1/20，因此E_{19}的最大误差为(1/21)×(1/20)= 0.0024。计算到E_{15}时，误差已减至少于4×10^{-8}，即比起始舍入误差还小许多，所以计算是稳定的。从E_{20}开始往前面计算，可有如下结果：

$E_{20}\approx 0.0$ $E_{19}\approx 0.0500000$ $E_{18}\approx 0.0500000$

$E_{17}\approx 0.0527778$ $E_{16}\approx 0.0557190$ $E_{15}\approx 0.0590176$

$E_{14}\approx 0.0627322$ $E_{13}\approx 0.0669477$ $E_{12}\approx 0.0717733$

$E_{11}\approx 0.0773523$ $E_{10}\approx 0.0838771$ $E_9\approx 0.0916123$

计算结果是可靠的。

例 1.18 对任意一个实数(甚至复数)x，e^x 可以表示成一个收敛的无穷级数：

$$e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$$

用基数$\beta=10$，$t=5$ 的浮点系计算$e^{-5.5}$。

解 在无穷级数 e^x 中，令 $x=-5.5$，有：

$$e^{-5.5}=1-5.5+15.125-27.730+38.129-41.942+38.446-30.208+20.768$$
$$-12.692+6.9803-3.4902+\cdots$$
$$e^{-5.5}=0.0026363$$

以上结果是取前面25项之和，第26项以后的各项不再改变这个和的大小了。然而事实上，$e^{-5.5}=0.00408677$。可见计算的结果没有一位有效。问题出在哪里呢？问题有两个：一是大数项的舍入误差太大了，像38.129这些项的舍入误差几乎和最后结果一样大了；二是由于舍弃了多位有效数字，使正负项发生了“灾难性抵消”，使得已经存在于各项的误差加以放大，以至产生灾难性后果。

虽然可以取更多的有效位来避免“灾难性抵消”，但这样做在时间和空间上是不划算的。更合理的解决办法是先计算 $x=5.5$ 的级数和，然后求它的倒数，即：

$$e^{-5.5}=\frac{1}{e^{5.5}}=\frac{1}{1+5.5+15.125+\cdots}=0.0040865 \quad (\text{用5位十进制运算})$$

这样计算的结果，其误差减少到0.007%。

由以上例子可知，对于一个实际问题，如果算法不适合将会得到不良的后果，而经过改善计算方法则可以避免出现不良的计算结果。因此，在构造一个具体数学问题的数值计算方法时，应该考虑计算方法的稳定性问题。除此之外，还要考虑收敛性问题（见第2章）等。

1.4 数值计算的基本原则

评价一个数值计算方法优劣的标准是：①计算时间复杂度（运算次数或计算时间，包括收敛性问题）；②计算空间复杂度（占用计算机存储空间）；③计算结果精确度（包括稳定性问题）。因此，要构造和选择一个好的计算方法，应该遵循如下一些基本原则：

1. 避免两个相近的数相减

在数值计算过程中，两个相近的数相减，会严重损失有效数字，从而使相对误差变大。

例1.18 计算 $9-\sqrt{80}$。

解 采用8位浮点数计算，其结果为：

$$\sqrt{80}=0.89442719\times10, \quad 9-\sqrt{80}\approx0.55728091\times10^{-1}$$

采用4位浮点数计算，其结果为：

$$\sqrt{80}=0.8944\times10, \quad 9-\sqrt{80}\approx0.5600\times10^{-1}$$

采用变换的公式计算（使用4位浮点数计算），其结果为：

$$9-\sqrt{80}=\frac{1}{9+\sqrt{80}}\approx0.5574\times10^{-1}$$

因此，在数值计算过程中两个相近的数相减，常常采用变换的公式进行计算。如：

$$\sqrt{x+\delta}-\sqrt{x}=\frac{\delta}{\sqrt{x+\delta}+\sqrt{x}}, \quad \frac{1-\cos x}{\sin x}=\frac{\sin x}{1+\cos x}, \quad \ln x-\ln y=\ln\frac{x}{y},$$

$$f(x+\delta)-f(x)=f'(x)\delta+\frac{1}{2}f''(x)\delta^2+\cdots$$

等等，如果计算公式不能改变，则可采用增加有效数字位数的方法来解决。

2. 避免使用绝对值很小的数作分母

用绝对值很小的数作除数,舍入误差会增大,而且当很小的数稍微有一点误差时,对计算的结果影响很大。例如:

$$\frac{3.81574}{0.0001}=38157.4,\quad \frac{3.81574}{0.0001+0.00001}=34688.5$$

计算结果的误差就很大。

3. 两个相差很大的数进行运算时,防止大数"吃掉"小数

例 1.19　计算 $x^2+(10^9+1)x+10^9=0$ 的根。

解　用因式分解法得到:$x_1=10^9, x_2=1$。

使用只能将数表示到小数点后 8 位的计算机,按求根公式

$$x_{1,2}=\frac{-b\pm\sqrt{b^2-4ac}}{2a}$$

进行计算。其结果为:

$$b=10^9+1=0.1\times10^{10}+0.0000000001\times10^{10}=0.1\times10^{10}$$

因为该计算机上只能表示到小数点后 8 位,计算机作加减法时要"对阶",对阶的结果是大数吃掉了小数,0.000000000×10^{10}在计算中不起作用。例如:

$$b^2-4ac\approx b^2\quad \sqrt{b^2-4ac}\approx|b|$$

得 $x_1\approx10^9$, $x_2\approx0$。

大数吃掉了小数在有些情况下是允许的,但在有些情况下则会产生谬误。理论分析表明,若干个数作加减运算时较小的数先运算,再与较大的数运算,其舍入误差最小。

4. 简化计算步骤,减少运算次数

例 1.20　请给出一种算法计算 2^{256},要求乘法次数尽可能少。

解　要尽可能减少运算量,其中一种思路就是,要尽可能运用已经计算出来的结果。

$$\begin{aligned}2^{256}&=2^{128}\times2^{128}=2^{64}\times2^{64}\times2^{128}=2^{32}\times2^{32}\times2^{64}\times2^{128}\\&=2^{16}\times2^{16}\times2^{32}\times2^{64}\times2^{128}=2^{8}\times2^{8}\times2^{16}\times2^{32}\times2^{64}\times2^{128}\\&=2^{4}\times2^{4}\times2^{8}\times2^{16}\times2^{32}\times2^{64}\times2^{128}\\&=2\times2\times2^{4}\times2^{8}\times2^{16}\times2^{32}\times2^{64}\times2^{128}\end{aligned}$$

这样共需 8 次乘法就可以计算出结果。

因此,简化计算步骤,减少运算次数,可以减少计算时间,提高计算精度。

本 章 小 结

本章重点论述了什么是数值计算方法,数值计算方法的研究内容,数值计算方法的特点和常用的算法,计算方法的稳定性和数值计算的基本原则,介绍了误差和有效数字等基本概念。通过本章学习,应该达到以下目标:

(1) 明确学习和掌握数值计算方法的重要意义;

(2) 深刻理解数值计算方法的理论性与技术性,以及面向计算机的内涵;

(3) 明确解决同一数学问题的数值计算方法并不是惟一的，在保证必要的计算精度的前提下，要选择空间复杂度和时间复杂度适合的计算方法；

(4) 在数值计算中，会正确应用近似计算的有关原则。

习　题　一

1.1　下列几个数都是经过四舍五入得到的近似值，试求各数的绝对误差、相对误差和有效数字的位数。

(1) 4580　(2) 0.0414　(3) 0.314×10^3　(4) 2789×10^{-2}

1.2　设有3个数 $a^*=333\pm1$，$b^*=3.33\pm0.01$，$c^*=0.00333\pm10^{-5}$，它们各自的近似值分别为 $a=333$，$b=3.33$，$c=0.00333$，试问哪个近似值的精度高？为什么？

1.3　设下列各对近似值均为有效数，试问它们是否一样？若不一样，有何区别？

(1) 45800 和 458×10^2；

(2) 0.00438 和 0.0438×10^{-1}；

(3) 0.4105×10^2 和 0.04105×10^3；

(4) 9800 和 98×10^2；

(5) 0.8070 和 0.807。

1.4　求 $\sqrt{10}$ 的近似值，使其相对误差不超过0.1%。

1.5　若1/4 用0.25 表示，问有多少位有效数字？

1.6　已知 $\pi=3.1415926\cdots$，取3.14 作为近似值，有3 位有效数字；取3.142 作为近似值，则有4 位有效数字。若取3.141 作为近似值，则可有几位有效数字？

1.7　试问6.1800 有几位有效数字？

1.8　某地粮食产量为875 万吨，可以表示为875 万吨 $=875\times10^4$ 吨 $=0.875\times10^7$ 吨，其绝对误差为1/2 万吨，或 $1/2\times10^4$ 吨，或 $1/2\times10^7$ 吨。试问875 万吨能不能表示为8750000 吨？为什么？

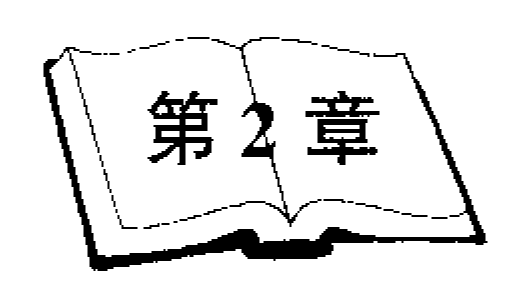
1.9　计算 $\frac{1}{662}-\frac{1}{663}$(保留4 位有效数字)。

1.10　如果计算 $a=(\sqrt{2}-1)^6$，当取 $\sqrt{2}=1.4$ 时，采用下列哪种方法好？

(1) $99-70\sqrt{2}$　　(2) $(3-2\sqrt{2})^3$　　(3) $\frac{1}{(3+2\sqrt{2})^3}$

(4) $\frac{1}{(\sqrt{2}+1)^6}$　　(5) $\frac{1}{99+70\sqrt{2}}$

1.11　古代数学家祖冲之曾以 $\frac{355}{113}$ 作为圆周率 π 的近似值，请问该近似值有多少位有效数字？



方程求根

2.1 引　　言

解代数方程是工程计算中经常遇到的数学问题。代数方程可以分类如下：

代数方程
- 单个方程
  - 线性方程
  - 非线性方程
    - 多项式方程
    - 超越方程
- 多个方程
  - 线性方程组（一组解）
  - 非线性方程组（多组解）

本章只讨论单个非线性方程 $f(x)=0$ 的求根问题。

方程的根可能是实（数）根也可能是复（数）根，可能是单根也可能是多重根。关于单根和重根，有如下定义：

定义2.1 对于方程 $f(x)=0$，如果 $f(x^*)=0$，但它的一阶导数 $f'(x^*)\neq 0$，则称 x^* 为方程 $f(x)=0$ 单根。推而广之，如果 $f(x^*)=f'(x^*)=f''(x^*)=\cdots=f^{(k)}(x^*)=0$，而 $f^{(k+1)}(x^*)\neq 0$，则称 x^* 为方程 $f(x)=0$ 的 $k+1$ 重根。

假设 $f(x)$ 有一单根 x^*，则 $f(x)$ 可写成 $f(x)=(x-x^*)g(x)$，且有 $g(x^*)\neq 0$。对 $f(x)$ 求导，得 $f'(x)=g(x)+(x-x^*)g'(x)$，故 $f'(x^*)=g(x^*)\neq 0$。类似地，可以推导出 k 重根的条件。

方程的根可能只有一个，也可能有几个或无穷多个，但也可能不存在。例如，对于多项式方程，其根的个数与方程的次数相同；对于超越方程，其根则可能是一个、几个或无穷多个，也可能不存在。因此在解方程时，有必要弄清楚方程根的性质。

此外，在实际问题中，有的问题要求解出方程所有的根，有的则只要找到其中某一个特殊根；有的要求将根的近似值逐步精确化，或者已知根所在的区间求此根；有的则只要求判断某个区域或范围内是否有根，而不要求把根求出来，等等。因此，也有必要弄清楚解方程的具体要求。总之，上述情况可归结为判断和确定根的分布区间这一重要问题。

2.2 方程根的分布区间

2.2.1 根的分布区间

定义2.2 若函数 $f(x)$ 在 $[a,b]$ 上连续，且 $f(a)\cdot f(b)<0$，方程 $f(x)=0$ 在 $[a,b]$ 内至少有一个根 x^*，则称 $[a,b]$ 为方程 $f(x)=0$ 的根 x^* 的分布区间；如果在 $[a,b]$ 内只有一个根 x^*，则称 $[a,b]$ 为方程 $f(x)=0$ 的根 x^* 的单根分布区间；如果在 $[a,b]$ 内只有一个根 x^*，并且始终有 $f'(x)\geqslant 0$ 或 $f'(x)\leqslant 0$，则称 $[a,b]$ 为方程 $f(x)=0$ 的根 x^* 的单调分布区间。

方程$f(x)=0$的根的分布情况可能会很复杂，当采用各种计算方法求方程的根时，都期望在较小的含根区间内进行，以保证根的惟一性和计算方法的收敛性或有效性。因此，在求方程的根之前，搜索方程根的分布区间，掌握方程根的分布情况是非常重要的。下面介绍几种通常采用的确定根的分布区间或单根分布区间的方法。

2.2.2 确定方程根的分布区间的方法

在工程计算中，确定方程根的分布区间的常用方法有草图法、搜索法和图像法。

1. 草图法

对于方程$f(x)=0$，先在函数$f(x)$的定义域内(任意)选取一些点，先计算出它们的函数值(在一般情况下这些函数值不会为零)，再根据这些已经计算出来的点，描绘出函数$f(x)$的图像，而函数$f(x)$的零点，即图像和坐标轴x的交点，就是方程$f(x)=0$的根。这种寻找函数零点的方法称为草图法。它适合人工计算且计算量不太大的场合，但在计算过程中，要保证所选取的自变量能使计算出的函数值有正负符号的变化。

当函数的性态(单调性变化、凸凹变化、拐点和极值点等)比较复杂时，用草图法一般难以精确表现出函数的图像，因而在搜索方程根过程中，可能会造成某个根的遗漏。因此，草图法只能对根的分布区间作大致判断。

例2.1 探索方程$f(x)=x^4-8.6x^3-35.51x^2+464.4x-998.46=0$的正实根的分布情况。

解 每隔一个单位计算一个函数值，可画出函数图像的略图，如图2.1所示。很显然，在$x=7$和$x=8$之间有一实根，但在靠近$x=4$的地方，可能有一重根(有一个切点)，或者有两个紧密间隔的实根，或者完全没有实根。事实上，通过计算发现，$x=4.3$是一重根。

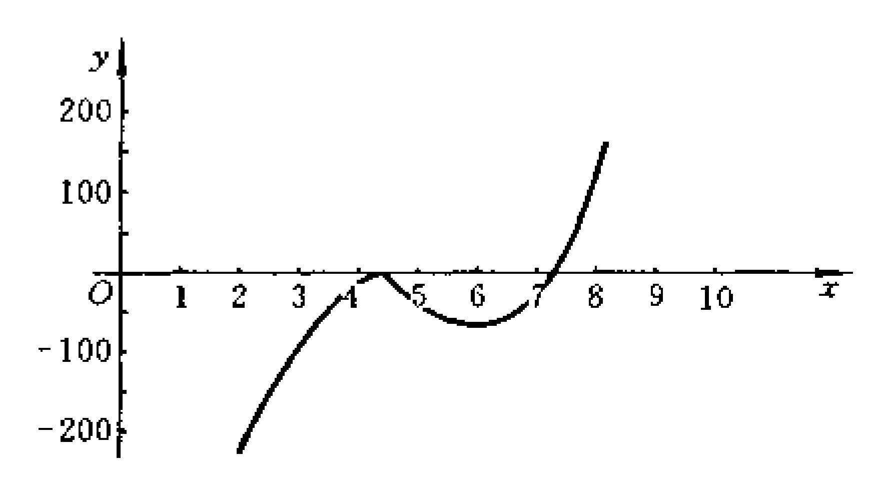


图2.1 例2.1的函数图像

2. 搜索法

对于方程$f(x)=0$，在函数$f(x)$的定义域内给定搜索起点x_0、搜索终点x_n和搜索步长h，利用计算机编程计算函数$f(x)$的一系列值。当相邻两点x_i和x_{i+1}的函数值出现正负符号变化，即$f(x_i)\cdot f(x_{i+1})<0$时，$[x_i,x_{i+1}]$就是方程$f(x)=0$的一个根的分布区间。这种方法称为搜索法。利用计算机计算速度快的特点，可以选取比较小的搜索步长h，从而可确定出方程所有根(不区分重根)的分布区间。

3. 图像法

图像法是指用数学软件，如MATLAB或Mathematica等来精确描绘函数图像的方法。图

像法能够准确、直观、全面地表示出方程根的分布情况和函数的性态，因此，图像法在求解方程根的过程中有着重要的作用。本书采用MATLAB语言编程来描绘函数的图像。例如，下面的程序fxg.m就是专门描绘函数图像的MATLAB程序：

```
fxg.m
function fxg(f_name,xmin,xmax,n_points)
f_name;
dx=(xmax-xmin)/n_points;
xp=xmin:dx:xmax;
yp=feval(f_name,xp);
plot(xp,yp,'r');                    % draw y=f(x)
xlabel('x');ylabel('f(x)');title('y=f(x)');hold on;
yp=0.0*xp;
plot(xp,yp)                         % draw x axis
```

使用程序fxg.m来描绘函数$y=f(x)$的图像时，应该首先编制用于定义$y=f(x)$的M函数，再按照要求的参数格式调用程序。

例2.2　用程序fxg.m描绘下列函数的图像：

(1) $y=x^3-x-1$；　(2) $y=\sqrt{1+x^2}-\text{tg}x$；　(3) $y=x(1-x\cos x)$。

解　(1) 编写f1.m函数如下：

```
function y=f1(x)
y=x.^3-x-1;
```

调用程序fxg.m的格式为fxg('f1',-2,2,500)，结果如图2.2(a)所示。

(2) 编写f2.m函数如下：

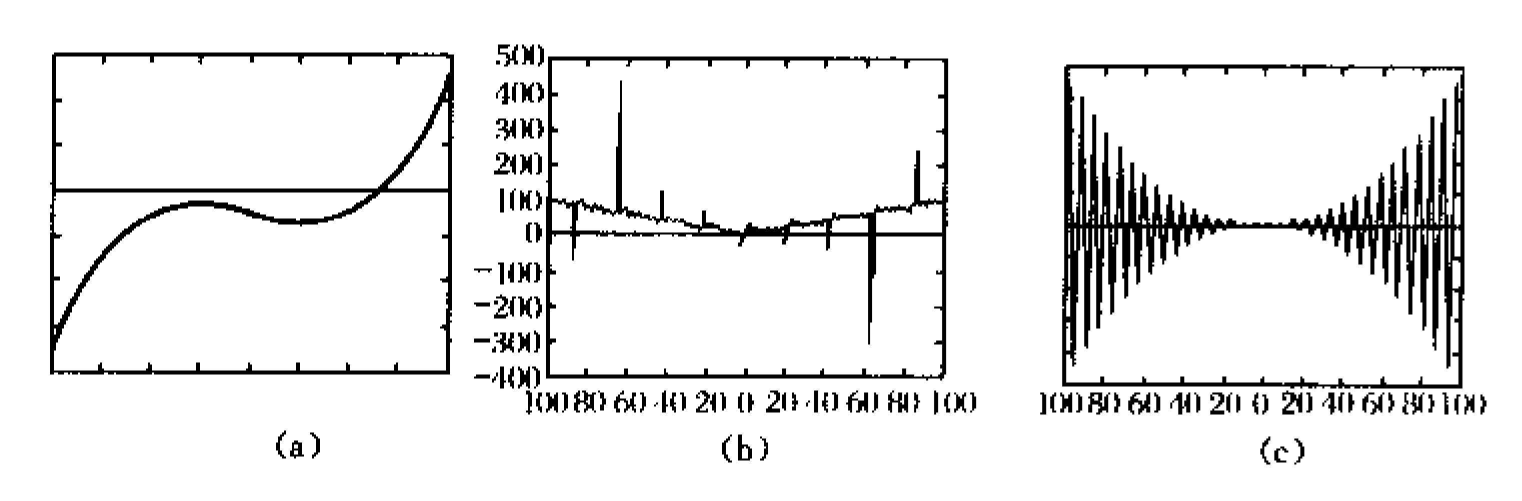
```
function y=f2(x)
y=sqrt(1+x.^2)-tan(x);
```

调用程序fxg.m的格式为fxg('f2',-100,100,5000)，结果如图2.2(b)所示。

(3) 编写f3.m函数如下：

```
function y=fx3
y=(1-x.*cos(x)).*x;
```

调用程序fxg.m的格式为fxg('fx3',-100,100,5000)，结果如图2.2(c)所示。



(a)　　(b)　　(c)

图2.2　例2.2的函数图像

根据上述函数图像，很容易看清楚例2.2函数的性态，从而能恰当地确定出方程根的分布区间。当函数比较复杂时，使用图像法格外方便和重要。

2.3 二分搜索法

所谓二分搜索法，就是将根的分布区间划分成两个部分，而方程的根必定会落在其中的一个部分或者就在划分点上，再对其中包含方程根的一个部分重复上述划分过程，从而不断缩小根的分布区间，直到根的分布区间的长度达到指定的精度要求为止。此时，根的分布区间内的任何一点，都是方程根的近似值。如果将根的分布区间划分成两个相等的部分，即将区间折半，这种二分法常常称为对分法。

在一般情况下，使用二分搜索法求方程的根，为了不漏掉方程的某些根，应该满足下面定理的条件。

定理 2.1 对实函数方程 $f(x)=0$，当 $x\in[a,c]$ 时，$f(x)$ 在 $[a,c]$ 上单调连续，且在两端点处为异号，即 $f(a)\cdot f(c)<0$，则方程在分布区间 $[a,c]$ 上只有一个根。

1. 用对分法求方程的单根

设方程 $f(x)=0$ 在区间 $[a_0,c_0]$ 上满足定理2.1的条件，且根的精确值为 x^*，计算误差为 ϵ。用对分法求方程根的过程如下：

1）第一次对分

（1）确定区间中点：将根的初始分布区间 $[a_0,c_0]$ 折半，得中点 $b_0=(a_0+c_0)/2$。

（2）确定含根区间：若 $f(a_0)\cdot f(b_0)=0$，则根 $x^*=b_0$，停止计算；若 $f(a_0)\cdot f(b_0)<0$，则根 x^* 在 b_0 的左侧，取 $a_1=a_0, c_1=b_0$，得到缩小的根的分布区间 $[a_1,c_1]$；若 $f(a_0)\cdot f(b_0)>0$，则根 x^* 在 b_0 的右侧，取 $a_1=b_0, c_1=c_0$，得到缩小的根的分布区间 $[a_1,c_1]$。

（3）确定区间大小：$[c_1-a_1]=[c_0-a_0]/2$。

（4）判断根的误差：取根的近似值为 $x_0=b_0=(a_0+c_0)/2$，与精确值 x^* 的误差为：

$$|e_0|=|x_0-x^*|\leqslant|c_1-a_1|=|c_0-a_0|/2$$

若 $|e_0|<\epsilon$，则停止计算，否则继续对分过程。

2）第二次对分

（1）确定区间中点：将根的分布区间 $[a_1,c_1]$ 折半，得中点 $b_1=(a_1+c_1)/2$。

（2）确定含根区间：若 $f(a_1)\cdot f(b_1)=0$，则根 $x^*=b_1$，停止计算；若 $f(a_1)\cdot f(b_1)<0$，则根 x^* 在 b_1 的左侧，取 $a_2=a_1, c_2=b_1$，得到缩小的根的分布区间 $[a_2,c_2]$；若 $f(a_1)\cdot f(b_1)>0$，则根 x^* 在 b_1 的右侧，取 $a_2=b_1, c_2=c_1$，得到缩小的根的分布区间 $[a_2,c_2]$。

（3）确定区间大小：$[c_2-a_2]=[c_0-a_0]/2^2$。

（4）判断根的误差：取根的近似值为 $x_1=b_1=(a_1+c_1)/2$，与精确值 x^* 的误差为：

$$|e_1|=|x_1-x^*|\leqslant|c_2-a_2|=|c_0-a_0|/2^2$$

若 $|e_1|<\epsilon$，则停止计算，否则继续对分过程。

3）继续对分计算

4）第 n 次对分

（1）确定区间中点：将根的分布区间 $[a_{n-1},c_{n-1}]$ 折半，得中点 $b_{n-1}=(a_{n-1}+c_{n-1})/2$。

（2）确定含根区间：若 $f(a_{n-1})\cdot f(b_{n-1})=0$，则根 $x^*=b_{n-1}$，停止计算；若 $f(a_{n-1})\cdot f(b_{n-1})$

<0,则根x^*在b_{n-1}的左侧,取$a_n=a_{n-1}$,$c_n=b_{n-1}$,得到缩小的根的分布区间$[a_n,c_n]$;若$f(a_{n-1})\cdot f(b_{n-1})>0$,则根$x^*$在$b_{n-1}$的右侧,取$a_n=b_{n-1}$,$c_n=c_{n-1}$,得到缩小的根的分布区间$[a_n,c_n]$。

(3) 确定区间大小:
$$[c_n-a_n]=\frac{[c_0-a_0]}{2^n} \tag{2.1}$$

(4) 判断根的误差:取根的近似值为:
$$x_{n-1}=b_{n-1}=\frac{(a_{n-1}+c_{n-1})}{2} \tag{2.2}$$

与精确值x^*的误差为:
$$|e_{n-1}|=|x_{n-1}-x^*|\leqslant[c_n-a_n]=\frac{[c_0-a_0]}{2^n} \tag{2.3}$$

若$|e_{n-1}|<\varepsilon$,则停止计算,否则继续对分过程,直到计算误差满足要求为止。

根据(2.3)式,可以确定满足计算精度要求的最小对分次数为:
$$n\geqslant\frac{\lg(c_0-a_0)-\lg\varepsilon}{\lg 2} \tag{2.4}$$

例如,如果$c_0-a_0=1$,$\varepsilon=0.0001$,则$n=14$。

2. 用二分法搜索方程的单实根

二分搜索法除了可用于求指定区间内(a,c)的根以外,还可用于将一个近似根逐步精确化;也可用于求方程$f(x)=0$在定义区间(a,c)内的所有单重实根。其方法如下:

自区间(a,c)的一个左端点a开始,并令$a_0=a$,且以某一个基本步长h,向右取一个宽度为h的小区间$(a_0,c_0=a_0+h)$,其中:

(1) 若$f(a_0)\cdot f(c_0)<0$,说明该小区间内存在一个实根,则用对分法找出这个实根,并且以该根为新端点继续往右搜索,依次再找出其他实根。

(2) 若$f(a_0)\cdot f(c_0)>0$,说明该小区间$(a_0,c_0=a_0+h)$中不存在实根,则继续向右再取一个步长为h的小区间$(a_0,c_0=a_0+2h)$,再次搜索。

这样,一个步长、一个步长地向右搜索,直至超过区间右端点c为止,即可找到方程定义区间(a,c)内所有单重实根。

显然,恰当地选择步长h是十分重要的。应使在所选择的一个步长内只有一个根,太大则会丢掉所要找的根,太小则会增加计算量。

二分搜索法除用于求方程的根之外,实际应用中,常用于查表,即在一组按大小顺序排列的数据中,寻找所需要的那个数。它比用顺序查表法的查表效率大大提高。

3. 对分法的 MATLAB 程序

在本书的附录A 的1.2 中,给出了对分法的MATLAB 程序bisec_g.m,它不仅给出了计算结果,而且同时图示了迭代过程。其调用格式为:

```
bisec_g('f_name',a,c,xmin,xmax,n_points)
```

其中,f_name 是用户编制的、用于定义f(x)的M 函数,a 和c 为初始区间的两个端点,xmin 与xmax 为图形中所能显示x的最小值和最大值,n_points 为描述图形所用的点数。容许的误差默认为$\varepsilon=10^{-6}$。

例 2.3 求下列两个函数的交点:

$$y=\sqrt{x^2+1}$$
$$y=\tan(x)\quad 0<x<\pi/2$$

解　该问题等价于求

$$f=\sqrt{x^2+1}-\tan(x)\quad 0<x<\pi/2$$

的零点，即解方程

$$\sqrt{x^2+1}-\tan(x)=0\quad 0<x<\pi/2$$

通过画出图形，可知该方程的根近似为 $x=0.9$。写出用于定义 $f(x)$ 的 M 函数为：

```
function f=fun_ex2(x)
f=sqrt(1+ x.^2)-tan(x);
```

然后调用 bisec_g.m 函数：

```
bisec_g('fun_ex2',0.8,1.0,0,1,300);
```

程序运行结果(图示过程省略)为：

f.name=

fun_ex2

Bisection Scheme

It	a	b	c	fa=f(a)	fc=f(c)	abs(fc-fa)	abs(c-a)/2
1	0.800000	0.900000	1.000000	0.250986	-0.143194	3.942e-001	1.000e-001
2	0.900000	0.950000	1.000000	0.085204	-0.143194	2.284e-001	5.000e-002
3	0.900000	0.925000	0.950000	0.085204	-0.019071	1.043e-001	2.500e-002
4	0.925000	0.937500	0.950000	0.035236	-0.019071	5.431e-002	1.250e-002
5	0.937500	0.943750	0.950000	0.008660	-0.019071	2.773e-002	6.250e-003
	……						
17	0.941461	0.941463	0.941464	0.000001	-0.000006	6.717e-006	1.526e-006
18	0.941461	0.941462	0.941463	0.000001	-0.000003	3.358e-006	7.629e-007

Tolerance is satisfied.

Finial result:Root=0.941462

例 2.4　求方程 $(1-x\cos x)x=0$ 在区间 $(2,6)$ 上的根。

解　首先编制用于定义 $f(x)$ 的 M 函数：

```
function y=dem_bs(x)
y=(1-x.*cos(x)).*x;
```

并为该 M 函数取名为 dem_bs.m，然后再调用 bisec_g 函数，具体格式为：

```
bisec_g('dem_bs',2,6,0,7,100)
```

程序运行结果为：

f.name=

dem_bs

Bisection Scheme

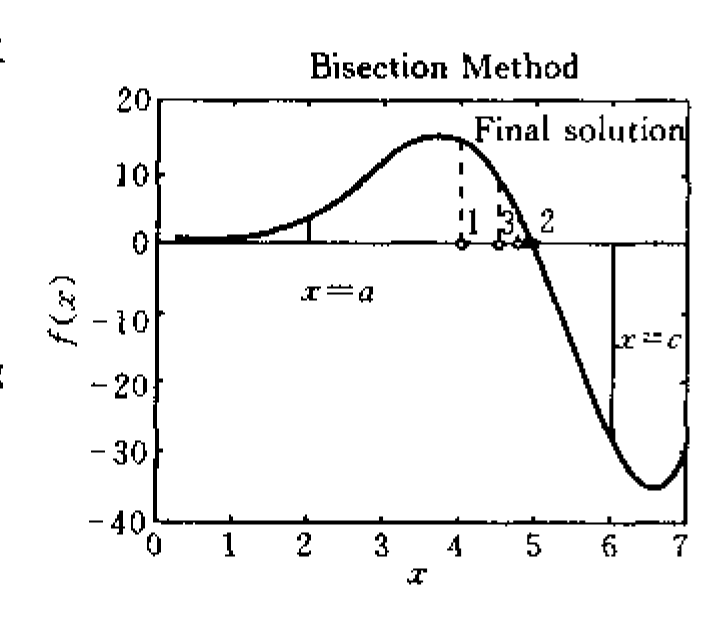


图 2.3　二分示意图

It	a	b	c	fa=f(a)	fc=f(c)	abs(fc-fa)	abs(c-a)/2
1	2.000000	4.000000	6.000000	3.664587	-28.566130	3.223e+001	2.000e+000
2	4.000000	5.000000	6.000000	14.458298	-28.566130	4.302e+001	1.000e+000
3	4.000000	4.500000	5.000000	14.458298	-2.091555	1.655e+001	5.000e-001
……							
21	4.917183	4.917185	4.917187	0.000074	-0.000020	9.412e-005	1.907e-006
22	4.917185	4.917186	4.917187	0.000027	-0.000020	4.706e-005	9.537e-007

Tolerance is satisfied.

Final result:Root=4.917186

经过$n=22$次计算,得到了方程的根$x=4.917186$。图2.3演示了$n\leqslant 3$时的对分计算过程。

使用对分法求方程的根,计算方法简单易用,只需知道方程根的单根分布区间即可。由公式(2.4)和上面的两个例题可知,使用对分法需要经过很多步的计算过程才能得到方程的近似根。为了加快方程的求根过程,有必要改进计算方法或采用其他较好的计算方法。

2.4 一般迭代法

2.4.1 基本原理和迭代公式

先看一个例子。设有两个函数$y=\varphi(x)$和$y=x$,欲求其交点x^*。为此,可将函数$y=x$改写成$x=y$的形式,并给定一个初始值x_0,并进行如下计算:

(1) 先计算函数$y=\varphi(x)$在x_0处的函数值y_0,然后计算函数$x=y$在y_0处的值x_1,即:

$$y_0=\varphi(x_0),\quad x_1=y_0$$

通过上述变换规则,由x_0得到x_1,x_1一般不同于x_0并且由x_0惟一确定。

(2) 再计算函数$y=\varphi(x)$在x_1处的函数值y_1,然后计算函数$x=y$在y_1处的值x_2,即:

$$y_1=\varphi(x_1),\quad x_2=y_1$$

通过变换规则,由x_1得到x_2,x_2不同于x_1并且由x_1惟一确定。

(3) 这样一直计算下去,就有可能会得到这样的结果:计算函数$y=\varphi(x)$在x^*处的函数值y^*,然后在计算函数$x=y$在y^*处的值时,得到的结果就是x^*,即:

$$y^*=\varphi(x^*),\quad x^*=y^*$$

即通过同样的变换规则,由x^*得到的仍然是x^*,并且该结果不再改变,x^*被称为不动点。上述按某种规则进行变换而得到某个数列$\{x_0,x_1,x_2,\cdots,x^*,\cdots\}$的过程称为迭代过程,即:

$$x_{k+1}=\varphi(x_k),\quad (k=0,1,2,\cdots)$$

在上述迭代计算过程中,如果数列$\{x_0,x_1,x_2,\cdots,x^*,\cdots\}$能趋向于某个极限值$x^*$,则这个极限值必然就是函数$y=\varphi(x)$和$x=y$的交点$x^*$,或者是方程组$y=\varphi(x)$和$x=y$的解。于是,就得到了一种解方程的数值方法——迭代方法。图2.4从几何上描绘了迭代的具体过程。

一般地,如果将方程组$y=\varphi(x)$和$y=x$消去y,则得到一个关于x的方程:

$$x=\varphi(x)$$

或

$$x-\varphi(x)=0$$

即

$$f(x)=0 \tag{2.5}$$

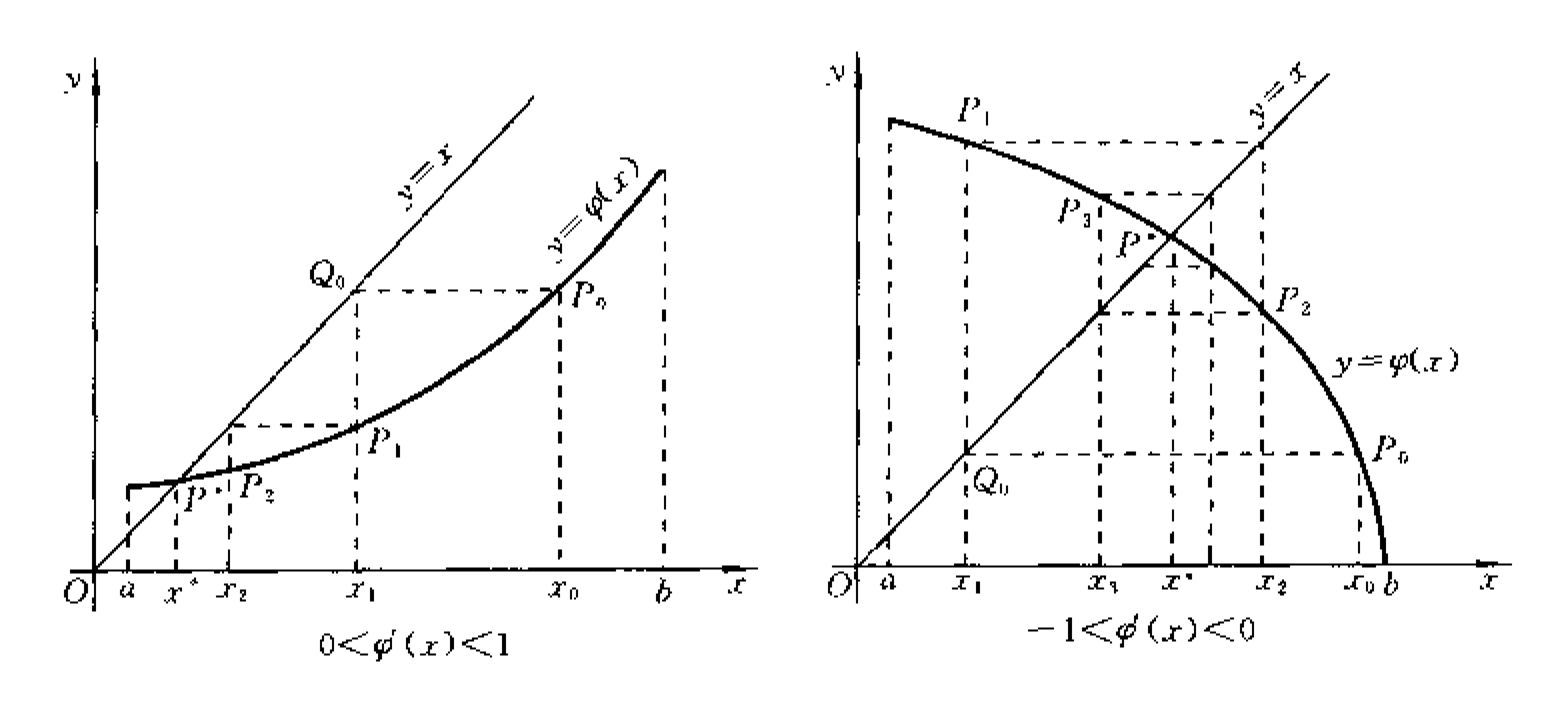


图2.4 迭代过程示意图

在方程$f(x)=0$中，两个函数的交点x^*必定满足$f(x^*)=0$，即$f(x)=0$和方程组$y=\varphi(x)$和$y=x$同解。根据这一思路，上溯推导，就可得到方程$f(x)=0$迭代求根的一般方法。即对一个方程

$$f(x)=0$$

进行等价变换，只要将其改写为隐式方程形式

$$x=\varphi(x) \tag{2.6}$$

就得到迭代的计算公式。并且，称$\varphi(x)$为迭代函数，称(2.6)式为一般迭代公式。由于$\varphi(x)$中仍然含有未知数x，故方程(2.6)不能直接求解。如果给出一个迭代初值x_0，则隐式方程(2.6)可转化为显示方程：

$$x_1=\varphi(x_0)$$

这样，就可以推导出一般迭代的格式：

$$x_{k+1}=\varphi(x_k),k=0,1,2\cdots \tag{2.7}$$

对于给定的迭代初值x_0，按迭代格式(2.7)反复计算，将得到一个序列$x_0,x_1,x_2,\cdots,x_k$。当$k\to\infty$时，如果极限$\lim\limits_{k\to\infty}x_k$存在，则称该迭代格式$x_{k+1}=\varphi(x_k)$是收敛的，并且，该极限就是方程$f(x)=0$的根，否则称该迭代格式$x_{k+1}=\varphi(x_k)$是发散的。

例2.5 求方程$x^3-x-1=0$在$x=1.5$附近的一个根。

解 可以将原方程改写为$x=\sqrt[3]{x+1}$的形式。由此，可写出一般迭代格式为：

$$x_{k+1}=\sqrt[3]{x_k+1},\quad k=0,1,2,\cdots$$

取迭代初值$x_0=1.5$，按照上述迭代格式进行初次迭代，得到$x_1=\sqrt[3]{x_0+1}=1.35721$，继续迭代下去，直到第7次和第8次迭代结果分别是$x_7=1.32472$和$x_8=1.32472$，两次迭代结果相同，便得到所求的根$x^*=1.32472$。表2.1列出了各次迭代的结果。

表2.1 例2.5各次迭代的结果

k	x_k	k	x_k	k	x_k
0	1.5	3	1.32588	6	1.32473
1	1.35721	4	1.32494	7	1.32472
2	1.33086	5	1.32476	8	1.32472

在使用一般迭代法的过程中,迭代计算结果并不总是收敛的。例如,如果将例2.5所给方程改写为$x=x^3-1$,其迭代格式为$x_{k+1}=x_k^3-1$。若迭代初值仍取$x_0=1.5$,则$x_1=2.375$,$x_2=12.3976$,…,这样继续迭代下去,显然得不到正确的结果,即当$k\to\infty$时不会趋于某个极限。这种迭代过程是发散的,而发散的迭代过程是毫无意义的。图2.5给出了迭代发散过程的图解(当$|\varphi'(x)|>1$时)。因此,在使用一般迭代法时,必须首先检验迭代格式的收敛性。

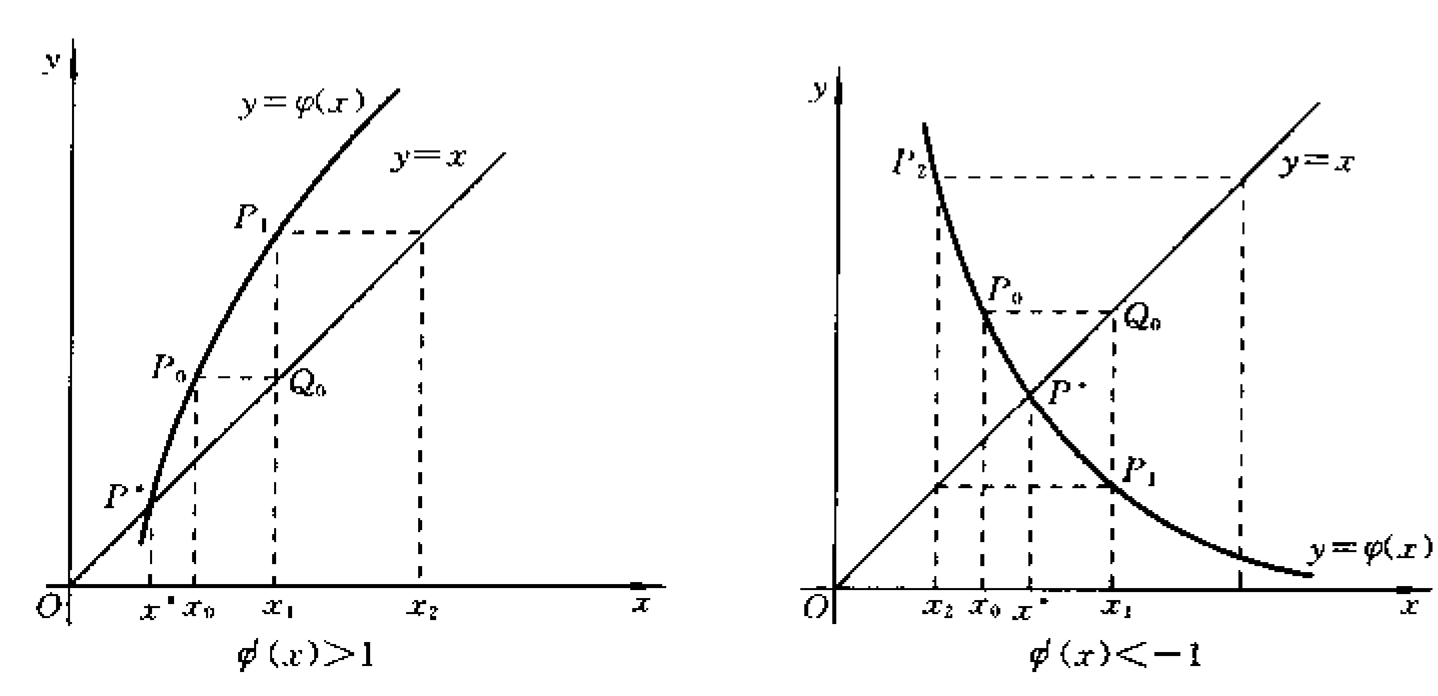


图2.5 迭代发散的过程

2.4.2 迭代法的收敛性

定理2.2 设函数$\varphi(x)$在$[a,b]$上具有连续的一阶导数,且满足条件:

(1) 对所有的$x\in[a,b]$,有$\varphi(x)\in[a,b]$;

(2) 存在$0<L<1$,对所有的$x\in[a,b]$,始终有$|\varphi'(x)|\leqslant L$;

则方程$f(x)=0$或$x=\varphi(x)$有如下性质:

(1) 在$[a,b]$上的解x^*存在且惟一;

(2) 对于任意的初始值$x_0\in[a,b]$,迭代过程$x_{k+1}=\varphi(x_k)$均收敛于x^*;

(3) 误差估计式为:

$$|x^*-x_k|\leqslant\frac{1}{1-L}|x_{k+1}-x_k|$$

或

$$|x^*-x_k|\leqslant\frac{L^k}{1-L}|x_1-x_0|\quad(k=1,2,\cdots)$$

证明 (1) 先证x^*的存在性。

在$[a,b]$上,$\varphi'(x)$存在,可导必然连续,故$\varphi(x)$连续。作函数

$$g(x)=x-\varphi(x)$$

则$g(x)$在$[a,b]$上亦连续。再由条件(1)得,$g(a)=a-\varphi(a)\leqslant0$,$g(b)=b-\varphi(b)\geqslant0$,故必然有$x^*\in[a,b]$,使得$g(x^*)=0$,即$x^*=\varphi(x^*)$成立。

(2) 再证x^*的惟一性。

若在$[a,b]$上另有$\bar{x}^*$也满足$\bar{x}^*=\varphi(\bar{x}^*)$,则由微分中值定理,有

$$x^*-\bar{x}^*=\varphi(x^*)-\varphi(\bar{x}^*)=\varphi'(\xi)(x^*-\bar{x}^*)$$

即

$$(x^*-\bar{x}^*)[1-\varphi'(\xi)]=0$$

其中ξ在x^*与x^*之间,故$\xi\in[a,b]$。由条件(2)知,$|\varphi'(\xi)|\neq1$,$1-\varphi'(\xi)\neq0$,所以只有$x^*=\bar{x}^*$

$=0$，即 $x^*=\bar{x}^*$。

(3) 再证明收敛性。

根据中值定理，有

$$x^*-x_{k+1}=\varphi(x^*)-\varphi(x_k)=\varphi'(\xi)(x^*-x_k),\quad (\xi \text{ 介于 } x^* \text{ 与 } x_k \text{ 之间})$$

再由条件(2) $|\varphi'(x)|\leqslant L$ 得

$$|x^*-x_{k+1}|=|\varphi'(\xi)(x^*-x_k)|\leqslant L|x^*-x_k|\quad (k=0,1,2,\cdots)$$

由该不等式递推，可得：

$$|x^*-x_k|\leqslant L|x^*-x_{k-1}|\leqslant L^2|x^*-x_{k-2}|\leqslant\cdots\leqslant L^k|x^*-x_0|$$

从而

$$\lim_{k\to\infty}|x^*-x_k|\leqslant\lim_{k\to\infty}L^k|x^*-x_0|$$

由 $0<L<1$，知 $\lim\limits_{k\to\infty}L^k\to 0$，故 $\lim\limits_{k\to\infty}x_k=x^*$。

(4) 最后证明误差估计式：

$$\begin{aligned}|x_{k+1}-x_k|&=|x^*-x_k-(x^*-x_{k+1})|\geqslant|x^*-x_k|-|x^*-x_{k+1}|\\&\geqslant|x^*-x_k|-L|x^*-x_k|=(1-L)|x^*-x_k|\end{aligned}$$

故

$$|x^*-x_k|\leqslant\frac{1}{1-L}|x_{k+1}-x_k|\quad (k=0,1,2,\cdots)$$

由于

$$|x_{k+1}-x_k|=|\varphi(x_k)-\varphi(x_{k-1})|=|\varphi'(\xi)(x_k-x_{k-1})|\leqslant|x_k-x_{k-1}|$$

其中，ξ 在 x_k 和 $x_{k-1}(k=0,1,2,\cdots)$之间。故有：

$$\begin{aligned}|x^*-x_k|&\leqslant\frac{L}{1-L}|x_{k+1}-x_k|\leqslant\frac{L}{1-L}|x_k-x_{k-1}|\\&\leqslant\frac{L^2}{1-L}|x_{k-1}-x_{k-2}|\leqslant\cdots\leqslant\frac{L^k}{1-L}|x_1-x_0|\end{aligned}$$

定理2.2即为一般迭代法的区间收敛性定理。在该定理中，条件(1)保证 $y=\varphi(x)$ 和 $y=x$ 一定有交点，条件(2)保证方程若有根则必然惟一并且能用迭代法求解出来。这两个条件是充分条件而不是必要条件。图2.6给出了定理2.2的几何解释。其中：(a)表示同时满足两个条件的情形，(b)表示条件(2)不满足的情形，(c)表示条件(1)不满足的情形，(d)表示条件(1)和条件(2)同时都不满足的情形。因此，在用迭代法求方程的根时，构造的迭代函数$\varphi(x)$必须同时满足两个条件。

定理2.2还给出了迭代过程的终止条件，即：

一个收敛的迭代过程，虽然从理论上讲有 $x^*=\lim\limits_{k\to\infty}x_k$，但在实际计算时，只能迭代有限次数，即只要迭代结果满足

$$|x_{k+1}-x_k|\leqslant\varepsilon \tag{2.10}$$

就能保证 x_k 充分接近方程的根 x^*。(2.10)式称为迭代过程的终止条件。

这里也可再来分析一下例2.5。由于同一方程一般可以构造出多个迭代函数，为了保证迭代过程收敛，如何确定收敛的迭代函数$\varphi(x)$以及如何选取迭代初始值x_0呢？根据定理2.2就能找到正确答案。

一般说来，定理2.2中的两个条件在较大的有根区间上是很难保证的，因此要尽可能寻找足够小的方程根的分布区间。通常可在根x^*附近来考察其收敛性。于是，就有下面的一般迭代法的局部收敛定理。

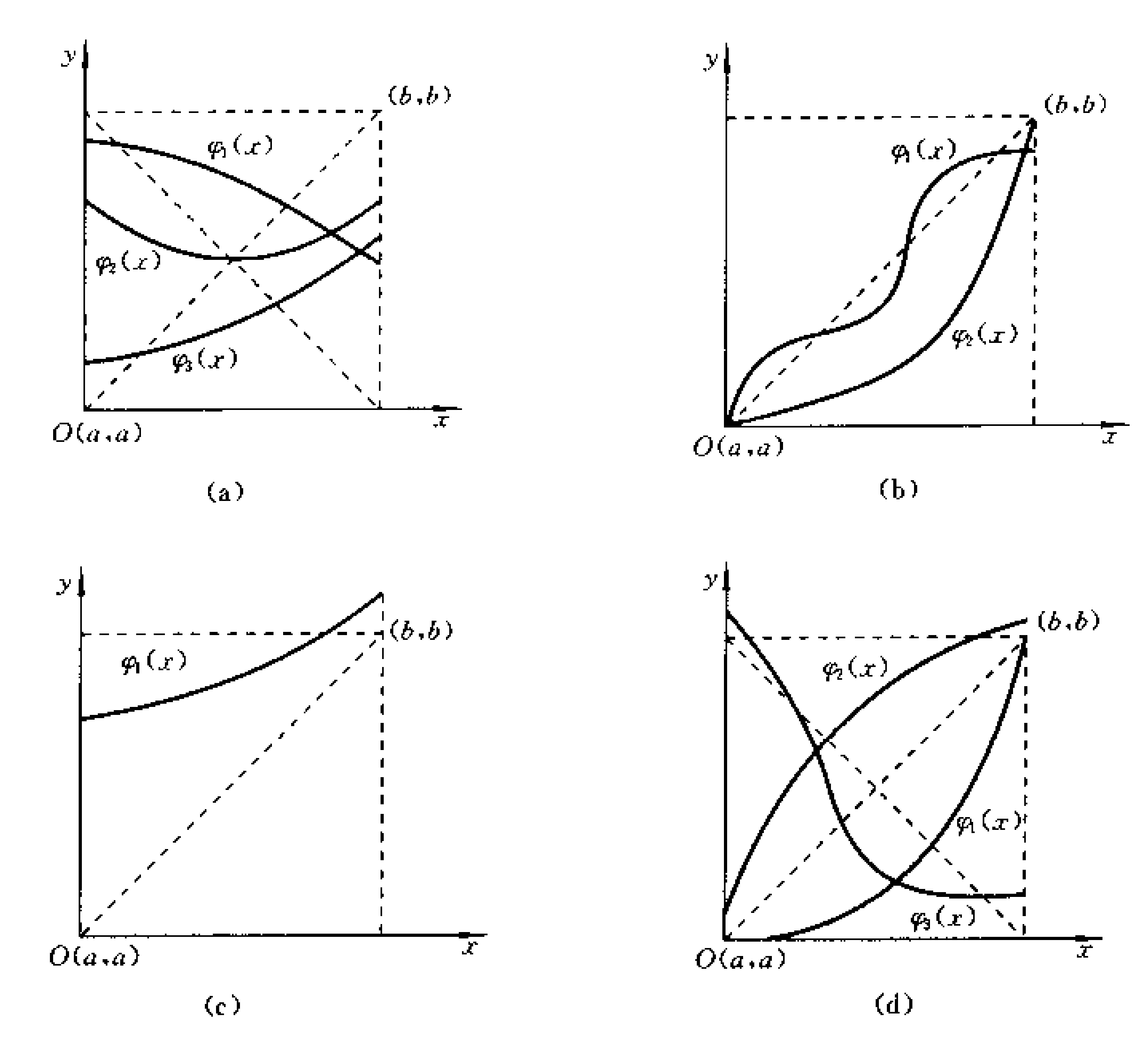


图 2.6　定理 2.2 的几何解释

定义 2.3　对于迭代函数 $\varphi(x)$，如果存在方程根 x^* 的一个邻域 $\Delta:|x-x^*|\leqslant\delta$，迭代过程对于任意初值 $x_0\in\Delta$ 均收敛，则称这种收敛性为局部收敛性。

定理 2.3　设 $\varphi(x)$ 在 $x=\varphi(x)$ 根 x^* 的邻域内有连续的一阶导数，且

$$|\varphi'(x^*)|<1 \tag{2.11}$$

则迭代过程具有局部收敛性。

证明　由于 $\varphi(x)$ 在 x^* 的邻域 $\Delta:|x-x^*|\leqslant\delta$ 内有连续的一阶导数，故根据微分中值定理，当 $x\in\Delta$ 时，有

$$\varphi(x)-\varphi(x^*)=\varphi'(\xi)(x-x^*)\quad(\xi\text{ 在 }x^*\text{ 和 }x\text{ 之间})$$

又因为 $|\varphi'(x^*)|\leqslant L<1$，所以

$$|\varphi(x)-\varphi(x^*)|=|\varphi'(\xi)(x-x^*)|\leqslant|x-x^*|\leqslant\delta$$

即当 $x^*-\delta\leqslant x\leqslant x^*+\delta$ 时，有 $\varphi(x^*)-\delta\leqslant\varphi(x)\leqslant\varphi(x^*)+\delta$，或 $x^*-\delta\leqslant\varphi(x)\leqslant x^*+\delta$，因此，根据定理 2.2，对于任意 $x_0\in\Delta$，迭代过程 $x_{k+1}=\varphi(x_k)$ 均收敛于 x^*。

例 2.6　用迭代法求方程 $2x^3-x^2-2=0$ 在[1,2]区间内的近似根。

解 1　确定迭代函数为：

$$\varphi_1(x)=\sqrt[3]{x^2/2+1}$$

选取计算初始值为 $x_0=1$，用 MATLAB 编程计算结果如下：

k	0	1	2	3	4	5	6	7	8
$x=\varphi_1$	1	1.1447	1.1829	1.1934	1.1963	1.1971	1.1973	1.1974	1.1974

解2　确定迭代函数为：

$$\varphi_2(x)=\sqrt{\frac{2}{2x-1}}$$

选取计算初始值为$x_0=1$，用MATLAB编程计算结果如下：

k	0	1	2	3	4	5	6	…	>60
$x=\varphi_2$	1	1.4142	1.0459	1.3535	1.0824	1.3103	1.1109	…	1.1974

解3　确定迭代函数为：

$$\varphi_3(x)=\frac{1}{2}+\frac{1}{x^2}$$

选取计算初始值为$x_0=1$，用MATLAB编程计算结果如下：

k	0	1	2	3	4	5	…	n	$n+1$
$x=\varphi_3$	1	1.5	0.9444	1.6212	0.8805	1.7899	…	0.5858	3.4142

用$x=\varphi_3(x)$迭代，计算结果为震荡发散。

用迭代法求方程的根，选用不同的迭代函数，所得到的收敛性和收敛速度是不一样的。因此，在实际计算中，应该选用收敛的迭代法以及收敛速度快的计算方法。

2.4.3　迭代法的收敛速度

迭代过程的收敛速度，是指迭代误差的下降速度。迭代法的收敛速度一般用收敛阶来描述。

定义2.4　对于收敛的迭代法$x_{k+1}=\varphi(x_k)$，$(k=1,2,\cdots)$，如果存在常数$p\geqslant1$，$c>0$，使得

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k^p}=C$$

成立(其中$e_k=|x_k-x^*|$)，则称该迭代法是p阶(次)收敛的。特别地，当$p=1$时称为线性收敛，$p=2$时称为平方收敛。

例2.7　讨论一般迭代法$x_{k+1}=\varphi(x_k)$，$(k=1,2,\cdots)$的收敛速度。

解　令$x^*=\varphi(x^*)$，所以$x_{k+1}-x^*=\varphi(x_k)-\varphi(x^*)$。根据中值定理，有

$$x_{k+1}-x^*=\varphi(x_k)-\varphi(x^*)=\varphi'(\xi)(x_k-x^*),\quad \xi\text{为}x_k\text{与}x^*\text{之间的某一点。}$$

因为$e_{k+1}=x_{k+1}-x^*$，$e_k=x_k-x^*$，所以当x_k在根x^*附近时，有$e_{k+1}=\varphi'(x^*)e_k$。

可见，当$\varphi(x^*)\neq0$时，一般迭代法$x_{k+1}=\varphi(x_k)$，$(k=1,2,\cdots)$，具有线性收敛性。

定理2.4　对于迭代过程$x_{k+1}=\varphi(x_k)$，如果迭代函数$\varphi(x)$在所求根x^*的邻近有连续二阶导数，且$|\varphi'(x^*)|<1$，则有：

(1) 当$\varphi'(x^*)\neq0$时，迭代过程为线性收敛；

(2) 当$\varphi'(x^*)=0$，而$\varphi''(x^*)\neq0$时，迭代过程为平方收敛。

证明　由已知条件知，迭代过程具有局部收敛性。

对于在根x^*附近收敛的迭代公式$x_{k+1}=\varphi(x_k)$，由于

$$e_{k+1}=x_{k+1}-x^*=\varphi(x_k)-\varphi(x^*)=\varphi'(\xi)(x_k-x^*)=\varphi'(\xi)e_k$$

这里ξ为x_k与x^*之间的某一点，当x_k在根x^*的附近时，将有$e_{k+1}\approx\varphi'(x^*)e_k$，故有：

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k}\to\varphi'(x^*)$$

因此，若$\varphi'(x^*)\neq 0$，则该迭代过程仅为线性收敛。若$\varphi'(x^*)=0$，则将$\varphi(x_k)$在根x^*处作泰勒展开，注意$\varphi'(x^*)=0$，有：

$$\varphi(x_k)=\varphi(x^*)+\frac{\varphi''(\xi)}{2!}(x_k-x^*)^2$$

又$\varphi(x_k)=x_{k+1}$，$\varphi(x^*)=x^*$，故由上式可得：

$$x_{k+1}-x^*=\frac{\varphi''(\xi)}{2!}(x_k-x^*)^2$$

从而

$$\lim_{k\to\infty}\frac{e_{k+1}}{e_k^2}\to\frac{\varphi''(x^*)}{2}\neq 0$$

迭代过程为平方收敛。

一般迭代法的收敛速度还可以是p阶收敛的。设$\varphi(x)$在$x=\varphi(x)$的根x^*附近有连续的p阶导数，且$\varphi'(x^*)=\varphi''(x^*)=\cdots=\varphi^{(p-1)}(x^*)=0$，$\varphi^{(p)}(x^*)\neq 0$，则迭代过程$x_{k+1}=\varphi(x_k)$是$p$阶收敛的。

在用迭代法求解方程的根时，可以先判断迭代函数的收敛速度然后再具体计算。作为练习，请判断例2.6中各种迭代函数的收敛速度。

2.4.4 收敛过程的加速

一个收敛的迭代过程，只要迭代次数足够多，就可以使计算结果达到任意指定的精度。但是，如果收敛过程过于缓慢、计算工作量过大，则在实际计算过程往往就要考虑加速收敛过程的问题。

1. 迭代公式的改进

设$\tilde{x}_{k+1}$表示由x_k经过一次迭代后所得到的结果$\tilde{x}_{k+1}=\varphi(x_k)$，由微分中值定理，有

$$x^*-\tilde{x}_{k+1}=\varphi(x^*)-\varphi(x_k)=\varphi'(\xi)(x^*-x_k)$$

其中，ξ为x_k与x^*之间的某一点。假设$\varphi'(x)$在求根范围内变化不大，可以近似地看成某个定值q，根据迭代法的收敛条件，有$|q|<1$。于是有：

$$x^*-\tilde{x}_{k+1}\approx q(x^*-x_k),\quad 从而\quad x^*\approx\frac{1}{1-q}\tilde{x}_{k+1}-\frac{q}{1-q}x_k$$

故

$$x_{k+1}=\tilde{x}_{k+1}+(x^*-\tilde{x}_{k+1})=\frac{1}{1-q}\tilde{x}_{k+1}-\frac{q}{1-q}x_k \tag{2.12}$$

x_{k+1}比$\tilde{x}_{k+1}$更接近根x^*。因此，迭代收敛的过程得到了加速。

构造一般迭代法加速公式的具体方法如下：

(1) $\tilde{x}_{k+1}=\varphi(x_k)$；

(2) $x_{k+1}=\tilde{x}_{k+1}+\frac{q}{1-q}(\tilde{x}_{k+1}-x_k)$，其中$|q|<1$。

2. Aitken(埃特金)加速法

在公式(2.12)中确定q时，需要计算迭代函数的导数$\varphi'(x)$。这在实际计算中是不太方便的，因此可以作进一步的改进。由于在上面已有：

$$\tilde{x}_{k+1}=\varphi(x_k),\quad x^*-\tilde{x}_{k+1}=q(x^*-x_k)$$

故将迭代值$\tilde{x}_{k+1}$再进行一次迭代计算，得：

$$\tilde{\tilde{x}}_{k+1}=\varphi(\tilde{x}_{k+1}),\quad x^*-\tilde{\tilde{x}}_{k+1}=q(x^*-\tilde{x}_{k+1})$$

所以
$$\frac{x^* - \tilde{x}_{k+1}}{x^* - \tilde{\tilde{x}}_{k+1}} = \frac{(x^* - x_k)}{(x^* - \tilde{x}_{k+1})}$$

解之，得
$$x^* \approx \tilde{\tilde{x}}_{k+1} - \frac{(\tilde{\tilde{x}}_{k+1} - \tilde{x}_{k+1})^2}{\tilde{\tilde{x}}_{k+1} - 2\tilde{x}_{k+1} + x_k} \tag{2.13}$$

于是，就有如下的迭代公式：

(1) $\tilde{x}_{k+1} = \varphi(x_k)$　　(2) $\tilde{\tilde{x}}_{k+1} = \varphi(\tilde{x}_{k+1})$　　(3) $x_{k+1} = \tilde{\tilde{x}}_{k+1} - \dfrac{(\tilde{\tilde{x}}_{k+1} - \tilde{x}_{k+1})^2}{\tilde{\tilde{x}}_{k+1} - 2\tilde{x}_{k+1} + x_k}$

Aitken 加速法的几何解释如图2.7所示。假设x_0为方程$x=g(x)$的一个近似根，由$x_1^{(1)}=g(x_0)$和$x_1^{(2)}=g(x_1^{(1)})$，在曲线上$y=g(x)$上可以定出两点$p_0(x_0, x_1^{(1)})$和$p_1(x_1^{(1)}, x_1^{(2)})$，作弦线$\overline{p_0p_1}$与直线$y=x$交于$p$，则$p$点坐标$x_1$满足：

$$x_1 = x_1^{(1)} + \frac{x_1^{(2)} - x_1^{(1)}}{x_1^{(1)} - x_0}(x_1 - x_1^{(1)})$$

解出x_1即得Aitken加速公式。

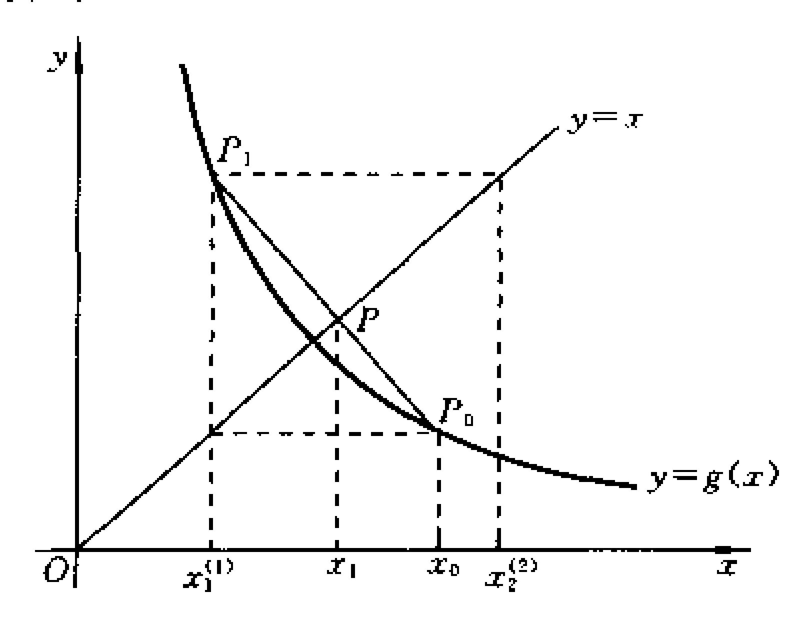


图2.7　Aitken 加速法的几何解释图

例2.8　用Aitken加速公式求解方程$x^3-x-1=0$。

解　由$x=\varphi(x)=x^3-1$，得：

$$\tilde{x}_{k+1} = x_k^3 - 1,\quad \tilde{\tilde{x}}_{k+1} = \tilde{x}_{k+1}^3 - 1$$

$$x_{k+1} = \tilde{\tilde{x}}_{k+1} - \frac{(\tilde{\tilde{x}}_{k+1} - \tilde{x}_{k+1})^2}{\tilde{\tilde{x}}_{k+1} - 2\tilde{x}_{k+1} + x_k}$$

取$x_0=1.5$，计算结果列于表2.2中。

此处将发散的迭代公式$x_{k+1}=x_k^3-1$，经过Aitken公式加速处理，获得了好的收敛性。

表2.2　例2.8的计算结果

n	$\tilde{x}$	$\tilde{\tilde{x}}$	x_k
0			1.5
1	2.37500	12.3965	1.41629
2	1.84092	5.23888	1.35565
3	1.49140	2.31728	1.32895
4	1.34710	1.44435	1.32480
5	1.32518	1.32714	1.32472

2.5　Newton(牛顿)法

2.5.1　基本思想与迭代公式

通常对已知方程$f(x)=0$进行变形而构造的迭代函数$\varphi(x)$不是惟一的。在实际应用中，如果希望迭代函数$\varphi(x)$有一种固定格式的构造方法，就可以采用下面的Newton迭代法。

Newton迭代法的基本思想是：设法将一个非线性方程$f(x)=0$转化为某种线性方程求解，其解决问题的基础是Taylor(泰勒)多项式。具体描述如下：

设$f(x)=0$的近似根为x_k，则函数$f(x)$在点x_k附近可用一阶Taylor 多项式$p_1(x)$来近似，即：

$$p_1(x)=f(x_k)+f'(x_k)(x-x_k)\cong f(x)$$

从而得到线性方程：

$$f(x_k)+f'(x_k)(x-x_k)=0 \tag{2.14}$$

解之，得该线性方程的根x，但它是$f(x)=0$的一个新近似根，记做x_{k+1}，即：

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)} \tag{2.15}$$

上式实质上就是一种迭代格式，称为Newton 迭代格式。相应地，Newton 迭代函数为：

$$\varphi(x)=x-\frac{f(x)}{f'(x)} \tag{2.16}$$

于是，按(2.16)式构造迭代函数解方程$f(x)=0$的方法，就是Newton 迭代法。

Newton 迭代法的几何解释如图2.8所示。方程$f(x)=0$的根x^*为$y=f(x)$和$y=0$(即x轴)的交点。设x_k为x^*的某个初始近似值，过p_k点$(x_k,f(x_k))$作$y=f(x)$的切线交x轴于x_{k+1}，即为所求得的近似值。继续过p_{k+1}点$(x_{k+1},f(x_{k+1}))$，p_{k+2}点$(x_{k+2},f(x_{k+2}))$，…，作$y=f(x)$的切线，即可逐步逼近精确的根x^*。因此，Newton 法也叫切线法，因为它是沿着曲线$y=f(x)$上某一点作切线逐步外推逼近的。从p_k点作切线与x轴的交点x_{k+1}，由于$y=f(x)$不是直线，所以$f(x_{k+1})$就不可能为零。因此必须以x_{k+1}作为新的起点，从与之对应的p_{k+1}点继续作切线，重复上述步骤，直至$f(x_{k+1})$充分小，逼近零时为止。

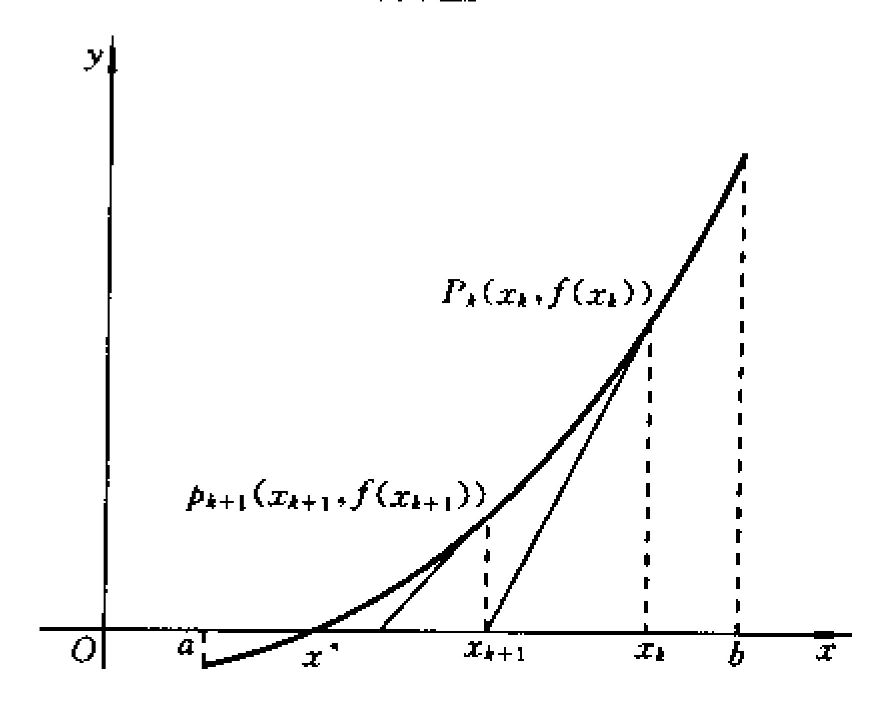


图2.8　Newton 迭代法的几何解释

例2.11　用Newton 迭代法求方程$xe^x-1=0$的根。

解　方程可改写为$x=e^{-x}$，即$f(x)=x-e^{-x}=0$，此方程的Newton 迭代格式为：

$$x_{k+1}=x_k-\frac{x_k-e^{-x_k}}{1+e^{-x_k}}=x_k-\frac{x_k-e^{-x_k}}{1+x_k}$$

取迭代初值为$x_0=0.5$，逐次迭代结果为$x_1=0.57102$；$x_2=0.56716$；$x_3=0.56714$；$x_4=0.56714$。由此可见，迭代4次即达到了$|x_4-x_3|\leqslant 0.000005$的要求，收敛速度是很快的。

例2.12　推导立方根的Newton 迭代公式，并求$a=155$的立方根。

解　该问题等价于求$f(x)=x^3-a$的零点，即解方程：

$$f(x)=x^3-a=0$$

运用Newton 迭代公式，有：

$$x_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}=x_k-\frac{x_k^3-a}{3x_k^2}=\frac{2}{3}x_k-\frac{a}{3x_k^2}$$

令 $a=155$，$x_0=5$，迭代计算结果为：

n	0	1	2	3
x	5	5.4	5.371834	5.371686(exact)

仅迭代3步就求得精确解。再使用较差一些的初值 $x_0=10$，则迭代计算结果为：

n	0	1	2	3	4	5
x	10	7.183334	5.790176	5.401203	5.371847	5.371686(exact)

迭代5步之后得到精确值。

使用Newton法求方程的根，在什么条件下、选取什么样的初始值 x_0，才能够保证迭代过程总是收敛于方程的根 x^* 呢？这是运用Newton法求方程根的重要问题。

2.5.2 Newton法的收敛性与收敛速度

1. 收敛性与收敛速度

对Newton法的迭代函数即公式(2.16)取导数，有：

$$\varphi'(x)=\frac{f(x)f''(x)}{[f'(x)]^2}$$

假定 x^* 是 $f(x)=0$ 的一个单根，即 $f(x^*)=0$，$f'(x^*)\neq 0$，则有 $\varphi'(x^*)=0$。所以，当 x 充分靠近 x^* 时，可使 $|\varphi'(x)|\leqslant L<1$ 成立，从而根据迭代法的局部收敛性定理可知，Newton迭代法在单根 x^* 的邻近是收敛的。即有下面的Newton法局部收敛性定理。

定理2.5 若 x^* 为方程 $f(x)=0$ 的一个单根，即 $f(x^*)=0$ 和 $f'(x^*)\neq 0$，$f(x)$ 在 x^* 邻域内有连续二阶导数，则由Newton公式(2.15)产生的迭代序列 $\{x_k\}$ 局部收敛于 x^*，并且收敛速度至少是平方收敛的(如果 $f''(x^*)\neq 0$，则收敛速度为平方收敛)。

这里只需证明收敛速度。

证明 因为 $f(x^*)=0$，$f'(x^*)\neq 0$，所以 $\varphi'(x^*)=0$，收敛速度至少是平方收敛的。

对 $\varphi'(x)=\dfrac{f(x)\cdot f''(x)}{[f'(x)]^2}$ 再求导一次，得：$\varphi''(x^*)=\dfrac{f''(x^*)}{f'(x^*)}$

所以，只要 $f''(x^*)\neq 0$，就有 $\varphi''(x^*)\neq 0$，因而根据定理2.4，收敛速度是平方收敛的。

可见，Newton法的收敛速度是较快的。定理2.5指出了Newton迭代法收敛的充分条件。在应用Newton法求方程的根时，应该在根 x^* 的附近选取迭代初值 x_0，以保证迭代过程的收敛性。在实际计算过程中，可以使用函数图像法来帮助确定迭代初值 x_0。如果要想知道Newton法在根的分布区间 $[a,b]$ 上的收敛性，则应该使用下面的定理。

定理2.6 设 $f(x)$ 在区间 $[a,b]$ 上二阶导数存在，且满足：

(1) $f(a)f(b)<0$

(2) $f'(x)\neq 0$，$x\in(a,b)$

(3) $f''(x)$ 不变号，$x\in(a,b)$

(4) $f''(x_0)f(x_0)>0$，$x_0\in(a,b)$

则Newton迭代公式(2.15)收敛于方程 $f(x)=0$ 在 (a,b) 内的惟一根 x^*。

图 2.9 给出了满足定理 2.6 条件的几种情况。定理 2.6 的条件(1)保证根的存在性;条件(2)表明函数单调,根惟一;条件(3)表明函数的凹凸性不变;条件(4)保证当 $x\in[a,b]$ 时,$\varphi(x)\in[a,b]$。

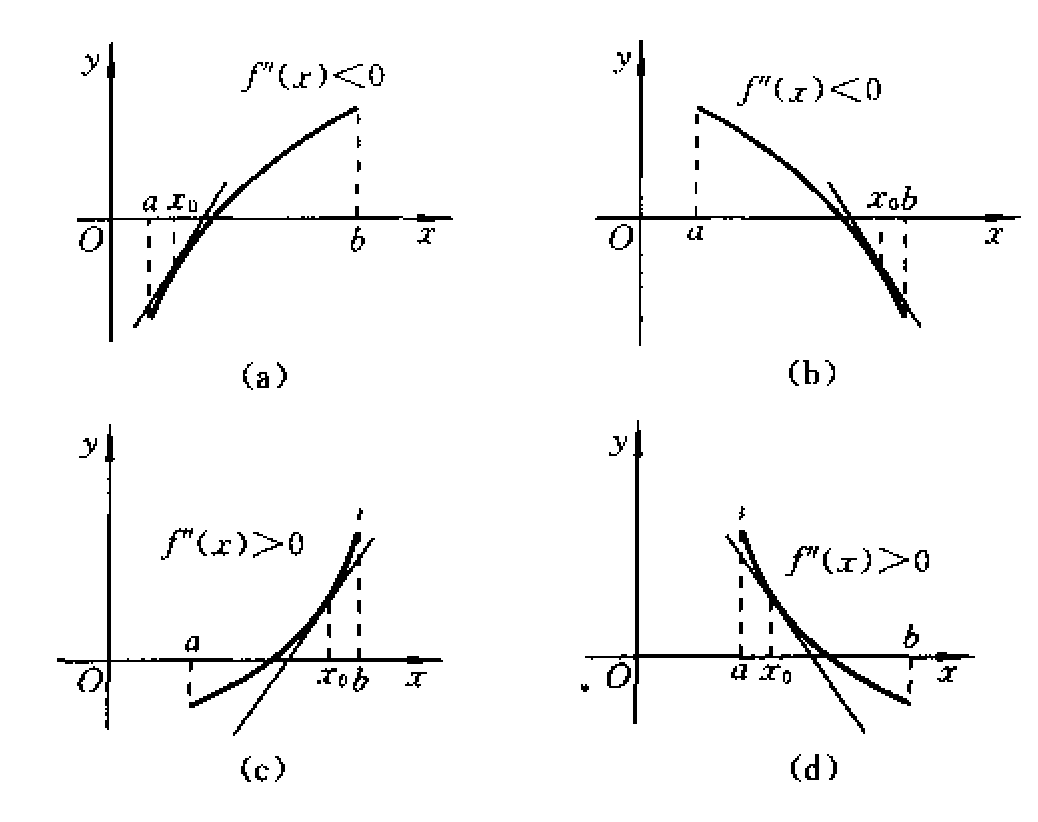


图 2.9 定理 2.6 的几何解释

定理 2.6 的条件(4)也可以表述为:

$$\frac{f(a)}{f'(a)}<b-a,\quad \frac{f(b)}{f'(b)}<b-a$$

此时,对任意的初始近似值 $x_0\in[a,b]$,Newton 法均收敛于方程的根 x^*。

例 2.13 求方程 $e^x-5x^2=0$ 在区间[2,5]上的根。

解 用图像法作出函数 $f(x)=e^x-5x^2$ 的图像,如图 2.10 所示。很显然,方程在区间[2,5]上有一单根。

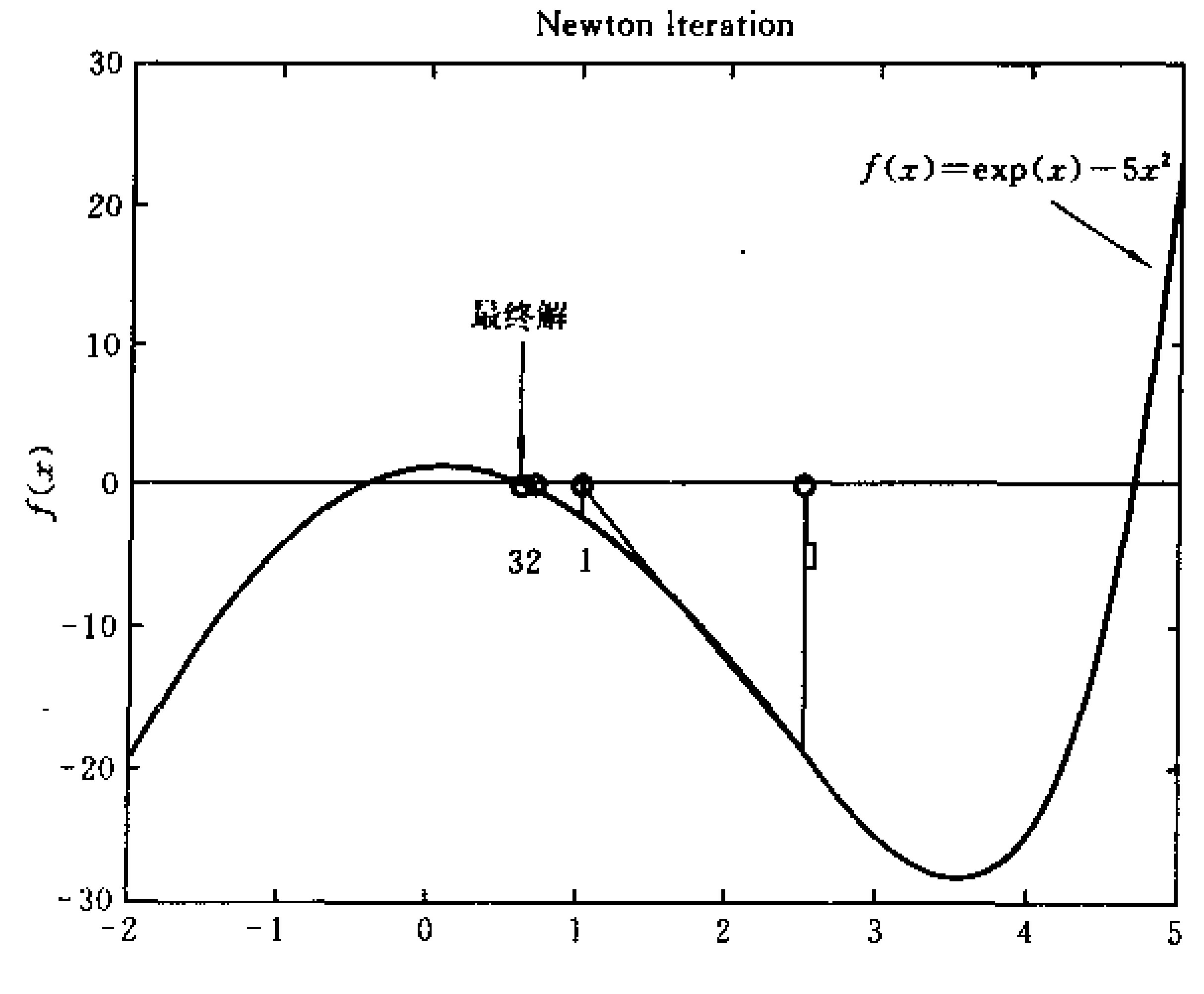


图 2.10 例 2.13 函数的图像

(1) 在根x^*的附近选取迭代初值，用局部收敛性定理来判断迭代过程的收敛性。其二阶导数$f''(x)=e^x-10$在整个定义域内为一连续函数，故Newton法在该方程（全部）根的附近（都）具有局部收敛性。选取初值$x_0=4.5$，计算结果为$x^*=4.7079379$。如果不是在区间[2,5]上靠近根的附近选取初值，就不能充分保证迭代过程收敛于根$x^*=4.7079379$。例如，选取初值$x_0=2.5$，则迭代计算结果收敛于方程的另外一个根$x_k=0.6052671$。

(2) 用区间收敛性定理来判断迭代过程的收敛性。

由$f'(x)=e^x-10x=0$，得到函数在区间[2,5]上的驻点$x_1=3.5771521$；由$f''(x)=e^x-10=0$，得到函数在区间[2,5]上的拐点$x_2=2.3025851$。因此，在区间$(x_1,5]$上：$f(x_1)f(5)<0$；对于任意的$x\in(x_1,5)$，始终有$f'(x)>0$和$f''(x)>0$。所以为满足$f''(x_0)f(x_0)>0$，必须使初值$x_0\in(x^*,5]$。如果即取$x_0=5$，则能充分保证迭代计算结果收敛于根x^*。但如果取$x_0=3.7\notin(x^*,5]$，$f''(x_0)f(x_0)>0$是否成立？这样做会有什么样的迭代计算结果？请动手试一试。

2. 初值的选取方法

应用Newton法求方程的根，选择初始值x_0很重要。如果x_0取得偏离所求方程的根较远，就可能使迭代过程发散或增加迭代次数。但是，根据Newton法收敛性定理来选取初始值，往往又比较复杂。为此，通常可以采用下面的简化方法：

对于方程$f(x)=0$，如果

$$f''(x_0)\neq 0,\quad |f'(x_0)|^2>\left|\frac{f(x_0)f''(x_0)}{2}\right| \tag{2.17}$$

则可以保证在大多数情况下Newton迭代过程的收敛性。

例2.14　用Newton法求$x^3-x-1=0$在$x_0=1.3$附近的一个根。

解　原方程的Newton迭代格式为$x_{k+1}=x_k-\dfrac{x_k^3-x-1}{2x_k^2-1}$，选取$x_0=1.5$，迭代结果为：

$$x_1=1.34783,\quad x_2=1.32520,\quad x_3=1.32472,\quad x_4=1.32472$$

即迭代4次，有6位有效数字。

如果迭代初值取为$x_0=0.6$，则迭代计算结果如下：

k	0	1	2	3	4	…	10	11
x_k	0.6	17.9	11.94680	7.985519	5.356908	…	1.324913	1.32472

即需要迭代11次才能达到计算精度的要求。

例2.15　求方程$f(x)=x^3-2x-5=0$在$x_0=2$附近的实根。

解　$[f'(2)]^2=[3\times 2^2-2]^2=10^2=100$，$|f(2)f''(2)|/2=1\times 12/2=6$，有：

$$[f'(2)]^2>|f(2)\cdot f''(2)|/2$$

所以可以用$x_0=2$作为初始值。迭代结果为$x_1=2.1$，$x_2=2.09457$，…，$x=2.09455$。

如果选取$x_0=-2$作为初值，虽然不能满足(2.17)式的条件，但计算结果还是收

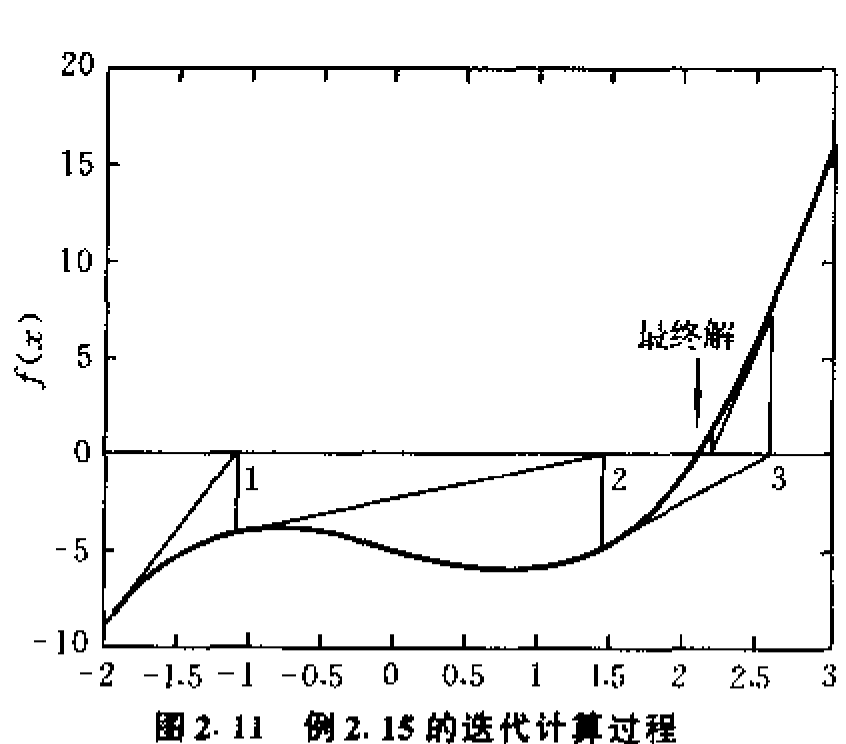


图2.11　例2.15的迭代计算过程

敛于方程的根,只不过是迭代次数有所增加。图2.11 表示了 $x_0=-2$ 的迭代计算过程。

2.6 Newton 迭代法的改进

2.6.1 Newton 下山法

Newton 法的收敛性与初值 x_0 的选取有很大关系,如果 x_0 偏离方程的根 x^* 太远,则 Newton 法可能发散或迭代次数增加,因此,有必要对迭代法加以改进。

例 2.16 用 Newton 法求 $x^3-x-1=0$ 在 $x_0=1.5$ 附近的一个根。

解 原方程的Newton 迭代格式为$x_{k+1}=x_k-\frac{x_k^3-x-1}{2x_k^2-1}$,迭代结果为$x_3=1.32472$。如果迭代初值取为 $x_0=0.6$,则迭代一次所得结果 $x_1=17.9$ 就大大远离了 x^*。因此,迭代过程可能是发散的或迭代次数增加。

为了避免迭代过程出现发散或迭代次数增加,可对迭代过程再附加如下一个条件:

$$|f(x_{k+1})|<|f(x_k)| \tag{2.18}$$

即保证函数值单调下降,则上例中的现象可以避免。

将条件(2.18)与Newton 迭代法合并起来,即在保证函数值稳定下降的条件下,用Newton 法来加快迭代,这就是 Newton 下山法。

Newton 下山法的具体操作如下:

将用Newton 法求得的结果$\overline{x_{k+1}}$与前一步迭代的近似值x_k适当加权平均,作为新的改进值 x_{k+1},即:

$$x_{k+1}=\lambda\overline{x_{k+1}}+(1-\lambda)x_k \tag{2.19}$$

或

$$x_{k+1}=x_k-\lambda\frac{f(x_k)}{f'(x_k)} \tag{2.20}$$

其中,$\lambda(0\leqslant\lambda<1)$称为下山因子。然后,适当选择$\lambda$的值,使之满足条件(2.18)。

下山因子λ的选择是一个逐步试探的过程,可以从$\lambda=1$ 开始,将λ逐步减半进行试算,一旦条件(2.18)满足,则称下山成功;否则称下山失败,需另选初值 x_0 再试。

现在用Newton 下山法来考察例2.16:仍取$x_0=0.6$,将用Newton 法求得的$\overline{x_1}=17.9$ 与初值 $x_0=0.6$ 加权平均,并取 $\lambda=\frac{1}{32}$,则

$$x_1=\frac{1}{32}\bar{x}_1+\frac{31}{32}x_0=1.140625$$

这个结果比17.9 要小许多。可见,上述严重偏离 x^* 的现象已经不复存在。

2.6.2 简化 Newton 法

应用Newton 法求方程的根,需要计算函数的导数。如果函数的导数太复杂,计算量大且繁琐,这时可以用常数值 c 来代替函数的导数值,则迭代公式就变成:

$$x_{k+1}=x_k-\frac{f(x_k)}{c} \tag{2.21}$$

这就是简化Newton 法。在(2.21)式中,常数值c 的选取可以根据迭代函数及收敛条件来确定。

记

$$\varphi(x)=x-\frac{f(x)}{c}$$

则
$$\varphi(x)=1-\frac{f'(x)}{c}$$

由 $|\varphi(x)|=\left|1-\frac{f'(x)}{c}\right|<1$，得
$$0<\frac{f'(x)}{c}<2 \tag{2.22}$$

这样，常数值 c 就容易确定了。

简化Newton法的几何意义是：过点 $(x_k, f(x_k))$，以 c 为斜率，作一直线与 x 轴的交点近似代替曲线与 x 轴的交点，如图2.12所示。

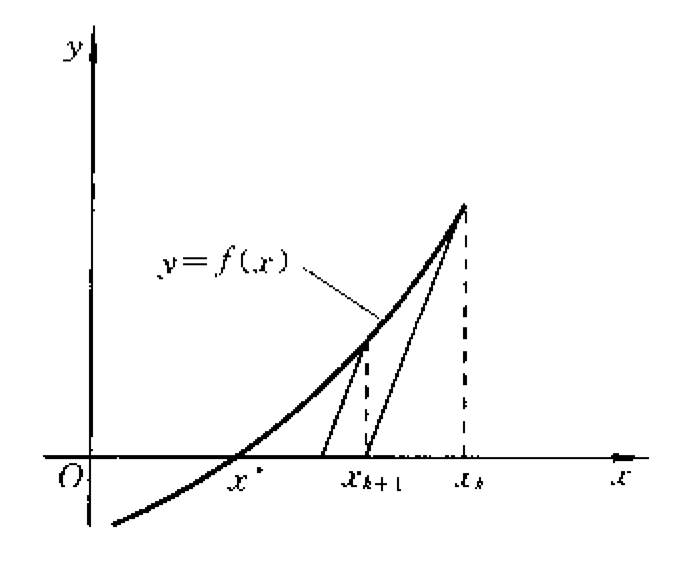


图2.12　简化Newton法

2.6.3　弦截法

弦截法是Newton法的简化与改进，它保留了Newton法收敛速度快的优点、克服了需要计算函数的导数 $f'(x)$ 的缺点。

如果用差商
$$\frac{f(x_k)-f(x_{k-1})}{x_k-x_{k-1}}$$
代替Newton迭代法中的 $f'(x_k)$，则可得到下面的离散化形式：
$$x_{k+1}=x_k-\frac{f(x_k)}{f(x_k)-f(x_{k-1})}(x_k-x_{k-1}) \tag{2.23}$$

(2.23)式中的等号右边，即为图2.13所示曲线 $y=f(x)$ 上 p_{k-1} 点 $(x_{k-1}, f(x_{k-1}))$ 与 p_k 点 $(x_k, f(x_k))$ 相连的直线或弦线的方程式。按(2.23)式求得的 x_{k+1}，就是该弦线与 x 轴的交点的横坐标，故该方法被称为弦截法。另一方面，对照Newton法，(2.23)式实际上也可以看成是用经过曲线 $y=f(x)$ 上 p_{k-1} 点 $(x_{k-1}, f(x_{k-1}))$ 和 p_k 点 $(x_k, f(x_k))$ 的割线的斜率来代替函数在 p_k 点 $(x_k, f(x_k))$ 的导数，故该方法又被称为割线法。

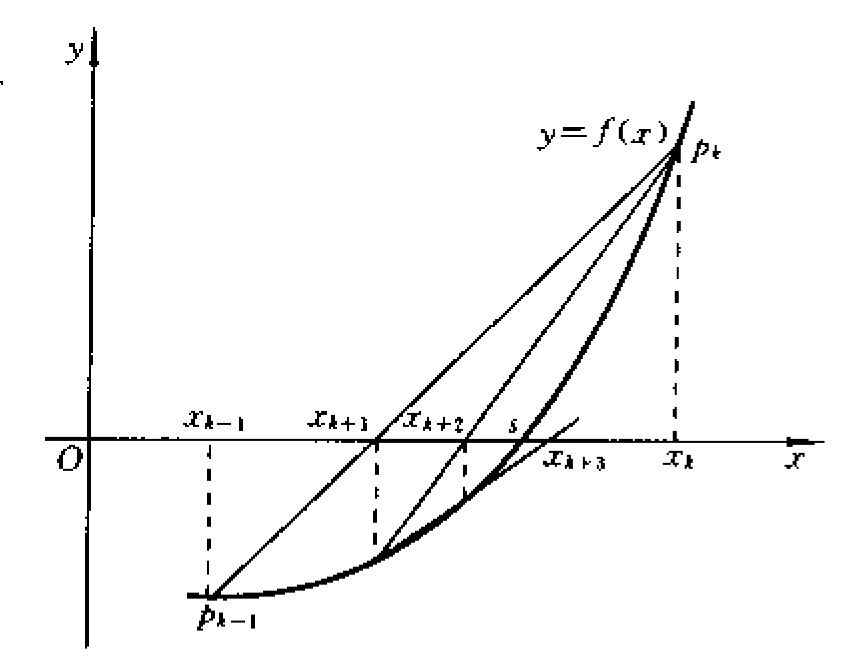


图2.13　双点弦截法

弦截法是一种两步迭代法，即在计算 x_{k+1} 时，必须先给出 x_{k-1} 和 x_k。因此，在使用公式(2.23)时，必须首先给出两个迭代初值 x_0 和 x_1，否则无法启动迭代运算。

在(2.23)式中，x_{k-1} 点可以一直用一个固定点 x_0 点来代替，此时弦截法演变成了单点弦截法，其几何解释如图2.14所示。相应地，图2.13称为双点弦截法。

弦截法的收敛性有下面两个定理：

定理2.7　设函数 $f(x)$ 在区间 $[a,b]$ 上存在二阶导数，且满足条件：

(1) $f''(x)$ 在区间 $[a,b]$ 上不改变符号；

(2) $f'(x)$ 在区间 $[a,b]$ 上不等于零；

(3) $f(a)f(b)<0$；

(4) $x_0\in[a,b]$ 且满足条件 $f(x_0)f''(x_0)>0$，$x_1\in[a,b]$。

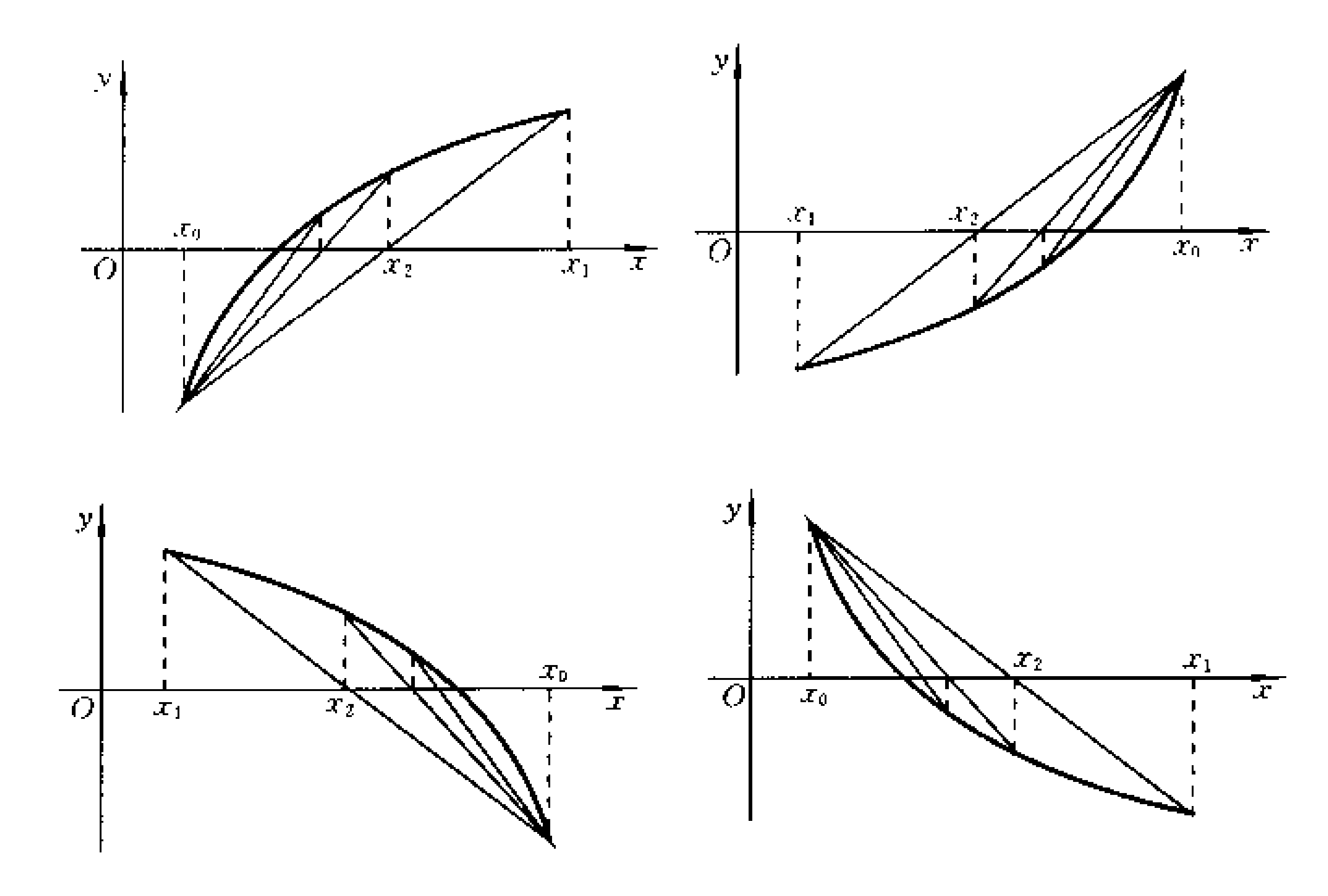


图 2.14 单点弦截法的几何解释

这时,由递推计算公式

$$x_{n+1}=x_n-\frac{x_n-x_0}{f(x_n)-f(x_0)}f(x_n),\quad (n=1,2,\cdots) \tag{2.24}$$

而得的序列$\{x_n\}$单调收敛于$f(x)$在$[a,b]$上的惟一解x^*。

该定理即为单点弦截法的收敛性定理。

定理 2.8 设x^*为$f(x)$的实根,$f'(x)$和$f''(x)$在包含x^*的某一区间是连续的,并且$f'(x^*)\neq 0$,则当x_0和x_1足够接近x^*时,由递推计算公式

$$x_{n+1}=x_n-\frac{x_n-x_{n-1}}{f(x_n)-f(x_{n-1})}f(x_n),\quad (n=1,2,\cdots) \tag{2.25}$$

而得的序列$\{x_n\}$收敛于x^*。

该定理即为双点弦截法的收敛性定理。

本 章 小 结

非线性方程求根的数值解法,首先要求确定根的分布区间,且具有局部收敛性的迭代格式其初始区间要尽可能小。在运用各种数值求解方法时,要考虑解的收敛性,进而要考虑收敛速度和计算量。本章要求掌握如下内容:

(1) 对于二分搜索法的解题思想,如何确定有根区间以及如何在容许误差范围内确定根的近似值。

(2) 对于简单迭代法的基本思想,如何构造迭代函数,如何判定迭代格式的收敛性。

(3) 对于牛顿迭代法的解题思想,牛顿迭代格式是如何建造的,熟悉迭代的计算过程和迭代初值对迭代速度的影响。

(4) 保证牛顿迭代法收敛性的条件,下山因子,牛顿下山法的迭代过程。

(5) 弦截法的迭代方法,弦截法可以避免计算导函数的原理,弦截法的迭代过程。

(6) 方程求根的MATLAB函数及其调用格式。

习 题 二

2.1 在区间$[0,\pi]$中用二分法计算$\tan^{-1}(3.5)$。提示：求解$\tan(x)=3.5, 0\leqslant x\leqslant \pi$。

2.2 对于用二分搜索法求方程$x^3-x-1=0$在$[1,2]$区间内的近似根，问：准确到10^{-3}至少要二分几次？

2.3 用二分法求解下列方程的根：

(1) $0.5\exp(x/3)-\sin(x)=0, x>0$

(2) $\log(1+x)-x^2=0$

2.4 方程$x^3-x^2-1=0$在$x=1.5$附近有根，可以把该方程写成如下4种不同的等价形式：

(1) $x=1+1/x^2$，对应的迭代格式为$x_{k+1}=1+1/x_k^2$

(2) $x^3=1+x^2$，对应的迭代格式为$x_{k+1}=\sqrt[3]{1+x_k^2}$

(3) $x^2=1/(x-1)$，对应的迭代格式为$x_{k+1}=\sqrt{1/(x_k-1)}$

(4) $x=\sqrt{x^3-1}$，对应的迭代格式为$x_{k+1}=\sqrt{x_k^3-1}$

试判断各个迭代格式在迭代初值$x_0=1.5$时的收敛性。选出其中一种迭代格式求方程的根到4位有效数字。

2.5 用Newton法求解下列方程：

(1) $f(x)=0.5\exp(x/3)-\sin(x), x>0$

(2) $f(x)=\log(1+x)-x^2$

(3) $f(x)=\exp(x)-5x^2$

(4) $f(x)=x^3+2x-1$

(5) $f(x)=\sqrt{x+2}-x$

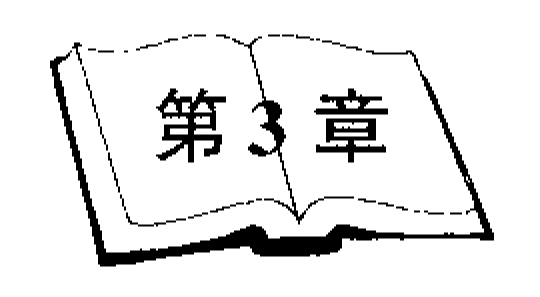
2.6 两个椭圆最多有4个交点，椭圆方程如下：

$$(x-2)^2+(y-3+2x)^2=5$$

$$2(x-3)^2+(y/3)^2=4$$

试用Newton下山法找出交点坐标。提示：先消掉x或y，只剩下一个变量之后再求解。

2.7 编制二分搜索法的计算流程图。



插值方法与曲线拟合方法

3.1 引　　言

在科学研究和工程实践中，要研究的函数 $y=f(x)$ 往往比较复杂，有时很难直接写出其数学表达式。但是，可以通过实验观测等手段，获得该函数的一组观测数据（离散采样值，或称采样点），如

$$(x_i, y_i) \quad (i=0,1,2,\cdots,n)$$

等，这样，函数 $y=f(x)$ 就直接以函数表的形式被表达出来了。显然，仅仅有函数表，研究和应用都很不方便。因此，希望用离散数据的函数表构造某个（简单的）函数 $g(x)$ 去逼近或代替原函数 $f(x)$。这种方法被称为数值逼近方法，其中，构造函数 $g(x)$ 称为逼近函数，原函数 $f(x)$ 称为被逼近函数。

常见的数值逼近方法有插值方法和曲线拟合方法等。

3.1.1 插值方法

如果观测数据的误差较小，则可将其作为准确值来处理，即观测数据 (x_i, y_i) 和函数 $f(x)$ 之间存在如下关系：

$$f(x_i)=y_i \quad (i=0,1,2,\cdots,n)$$

因此，要构造的函数 $g(x)$ 也应该严格满足下列关系：

$$g(x_i)=y_i \quad (i=0,1,2,\cdots,n)$$

即函数 $g(x)$ 通过所有的采样点。这就给出了函数 $g(x)$ 的一种构造约束条件。

定义3.1 已知函数 $y=f(x)$ 定义在区间 $[a,b]$ 上，在已知点 $a\leqslant x_0<x_1<x_2<\cdots<x_n\leqslant b$ 处的值为 $y_0, y_1, y_2, \cdots, y_n$。如果存在一个函数 $p(x)$ 满足插值条件，即

$$p(x_i)=y_i \quad (i=0,1,2,\cdots,n) \tag{3.1}$$

则称 $p(x)$ 为 $f(x)$ 的插值函数，$x_0, x_1, x_2, \cdots, x_n$ 为插值节点，区间 $[a,b]$ 称为插值区间，任意给定的点 x 称为插值点，构造插值函数 $p(x)$ 的方法称为插值方法。特别地，若插值函数 $p(x)$ 是一个次数不超过 n 的代数多项式

$$p_n(x)=a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \quad (a_i \text{ 为实数})$$

则称 $p_n(x)$ 为插值多项式。这种插值方法称为多项式插值方法。

一般地，插值方法适用于观测数据的准确度和可靠性较高的场合。

3.1.2 曲线拟合方法

如果观测数据的误差较大，不可能也没有必要将它作为准确值来处理，则只要求所构造的函数 $g(x)$ 尽可能靠近这些采样点而不要求通过这些采样点。即要求所构造的函数 $g(x)$ 和采样

点$(x_i,y_i)(i=1,2,\cdots,n)$之间满足某种误差准则，如

$$Q=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{n}[(y_i-g(x_i)]^2$$

等，使总的偏差Q为最小即可。按照上述要求构造函数$g(x)$的方法，就是曲线拟合方法，其中函数$g(x)$称为拟合函数。

可见，曲线拟合方法适用于观测数据本来就含有不可避免误差的场合，不要求构造的函数$g(x)$严格地通过采样点(x_i,y_i)，而只要求尽可能地靠近它们。为了提高拟合精度，要求提供尽可能多的采样点或观测值。

3.2 Lagrange(拉格朗日)插值法

Lagrange 插值法是一种多项式插值方法。下面先从两点插值、三点插值等简单情况开始讨论，然后推广到多点插值即Lagrange 插值的讨论。

3.2.1 线性插值(两点插值或一次插值)

线性插值就是通过两个采样点(x_0,y_0)和(x_1,y_1)，作一直线$p_1(x)$来近似代替$f(x)$。根据插值条件(3.1)，有：

$$p_1(x_0)=y_0,\quad p_1(x_1)=y_1$$

因此，可以写出直线$p_1(x)$的以下两种表达式：

(1) 点斜式： $$p_1(x)=y_0+\frac{y_1-y_0}{x_1-x_0}(x-x_0)$$

(2) 对称式： $$p_1(x)=\frac{x-x_1}{x_0-x_1}y_0+\frac{x-x_0}{x_1-x_0}y_1$$

在点斜式中，$\frac{y_1-y_0}{x_1-x_0}$即为差商，即当$x_1\to x_0$时，它就是y_0'。将其代入点斜式方程中，可得到$p_1(x)$的极限形式：

$$p_1(x)=y_0+y_0'(x-x_0)$$

这就是一阶Taylor(泰勒)多项式。在这里，$p_1(x)$由两项组成：一项是x的零次多项式，另一项为x的一次多项式。$p_1(x)$的这种组成形式就是后面将要介绍的Newton(牛顿)插值公式的形式。

在对称式中，若令

$$\frac{x-x_1}{x_0-x_1}=l_0(x),\quad \frac{x-x_0}{x_1-x_0}=l_1(x) \tag{3.2}$$

则有

$$p_1(x)=l_0(x)\cdot y_0+l_1(x)\cdot y_1 \tag{3.3}$$

(3.2)和(3.3)式就是线性插值公式。其中，$l_0(x)$和$l_1(x)$是关于x的线性多项式，称之为插值基函数。它们在节点x_0和x_1处分别满足：

$$l_0(x_0)=1,\quad l_0(x_1)=0$$
$$l_1(x_0)=0,\quad l_1(x_1)=1$$

于是，可得出重要结论：满足插值条件$p_1(x_0)=y_0$，$p_1(x_1)=y_1$的一次插值多项式$p_1(x)$，可用两个插值基函数$l_0(x)$和$l_1(x)$进行线性组合构造。即

$$p_1(x)=l_0(x)\cdot y_0+l_1(x)\cdot y_1$$

例 3.1 已知 $y=\sqrt{x}$ 的两个采样点(100,10)和(121,11),求 $\sqrt{115}$。

解 用点斜式计算:

$$p_1(\sqrt{115})=10+\frac{11-10}{121-100}(115-100)=10.714286$$

用对称式计算:

$$p_1(\sqrt{115})=\frac{115-121}{100-121}\times 10+\frac{115-100}{121-100}\times 11=10.714285$$

与精确值 10.7238 相比,两者均有 3 位有效数字。

在MATLAB 中,有一个函数interp1 可以对函数表进行线性插值,确定横坐标为xi 的指定点的函数。函数(命令)interp1 在每一段数据区间上做线性插值,其中 xi 也可以是一个用来指定x 值的向量,它的调用格式是:

yi=interp1(x,y,xi)

这里x 表示数据横坐标值的列数组(x 必须为单调),y 表示数据纵坐标值的列数组。两个列数组的长度必须一致,不过 y 可以包含多列。xi 是一个标量或表示 x 值的数组,其对应的 y 值通过线性插值算出。

例 3.2 假设给出如下表格形式的函数关系 $y=y(x)$:

x	0	0.25	0.50	0.75	1.00
y	0.9162	0.8109	0.6931	0.5596	0.4055

其中,$y(x)$是关于x 的单调增函数。用MATLAB 分别计算出使$y=0.9,0.7,0.6$ 和0.5 的x 值。

解 这是一个反插值问题,就是说将x 看做y 的函数,即$x=f(y)$。解此问题的MATLAB程序如下:

```
    x=[0.0,0.25,0.5,0.75,1.0 ];
    y=[0.9162,0.8109,0.6931,0.5596,0.4055];
    yi=[0.9,0.7,0.6,0.5]';
    xi=interp1 (y,x,yi);
    [yi,xi]
%      给出结果,第 1 列是y 值,第 2 列是x 值
ans=
    0.9000   0.0385
    0.7000   0.4854
    0.6000   0.6743
    0.5000   0.8467
```

3.2.2 抛物插值(三点插值或二次插值)

抛物插值就是通过 3 个采样点(x_0,y_0),(x_1,y_1)和(x_2,y_2)构造一个二次多项式 $p_2(x)$来近似代替$f(x)$。根据插值条件(3.1),有:

$$p_2(x)=y_i \quad (i=0,1,2)$$

因此,根据插值条件,用待定系数法可以确定出二次多项式 $p_2(x)=a_0+a_1x+a_2x^2$ 的各个系数。这里为避免解方程组,使用基函数线性组合的构造方法来求二次多项式$p_2(x)$。由线性插

值的结论推广可知，该二次多项式$p_2(x)$可用3个插值基函数$l_0(x)$，$l_1(x)$和$l_2(x)$进行线性组合构造。即：

$$p_2(x)=l_0(x)\cdot y_0+l_1(x)\cdot y_1+l_2(x)\cdot y_2 \tag{3.4}$$

3个插值基函数在插值节点x_0，x_1和x_2处应该分别满足：

$$\begin{array}{lll} l_0(x_0)=1 & l_0(x_1)=0 & l_0(x_2)=0 \\ l_1(x_0)=0 & l_1(x_1)=1 & l_1(x_2)=0 \\ l_2(x_0)=0 & l_2(x_1)=0 & l_2(x_2)=1 \end{array}$$

即只要确定出3个插值基函数即可。

根据$l_0(x_1)=l_0(x_2)=0$，可假设$l_0(x)=C(x-x_1)(x-x_2)$；将$l_0(x_0)=1$代入，得：

$$C(x_0-x_1)(x_0-x_2)=1$$

即

$$C=\frac{1}{(x_0-x_1)(x_0-x_2)}$$

所以

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} \tag{3.5}$$

同理

$$l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \tag{3.6}$$

$$l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} \tag{3.7}$$

这样，由(3.4)、(3.5)、(3.6)和(3.7)式就构成了抛物插值公式。即：

$$p_2(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}\cdot y_0+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}\cdot y_1+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}\cdot y_2$$

于是，可得出重要结论：抛物插值公式由3个二次插值基函数线性组合而成。

例3.3　已知$y=\sqrt{x}$在节点110，121和144处的值为10，11和12，试构造一个二次插值多项式$p_2(x)$，并求$\sqrt{115}$的近似值。

解　先求出3个插值基函数的值：

$$l_0(x)=0.18831169 \qquad l_1(x)=0.9006211 \qquad l_2(x)=-0.0889328$$

所以 $\sqrt{115}\cong p_2(\sqrt{115})=0.18821169\times10+0.90062\times11-0.0889328\times12=10.722755$

这个结果与精确值10.7238相比，具有4位有效数字。可见，抛物插值要比线性插值精确。

3.2.3 Lagrange 插值

Lagrange插值法就是通过多个采样点$(x_i,y_i)(i=0,1,2,\cdots,n)$构造一个高次多项式$p(x)$来近似代替$f(x)$。关于插值节点数和多项式次数之间的关系，有如下定理：

定理3.1　在$n+1$个互异的插值节点$x_0,x_1,x_2,\cdots,x_n$上，满足插值条件(3.1)式并且次数不高于n的代数多项式$p_n(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$存在且惟一。

证明　根据插值条件，有：

$$\begin{cases} p_n(x_0)=a_0+a_1x_0+a_2x_0^2+\cdots a_nx_0^n=y_0 \\ p_n(x_1)=a_0+a_1x_1+a_2x_1^2+\cdots a_nx_1^n=y_1 \\ \quad\vdots \\ p_n(x_n)=a_0+a_1x_n+a_2x_n^2+\cdots a_nx_n^n=y_n \end{cases}$$

该式是一个关于未知数$a_0,a_1,a_2,\cdots,a_n$的线性方程组，其系数矩阵的行列式是

Vandermonde(范德蒙)行列式:

$$V=\begin{vmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{vmatrix}=\prod_{i=1}^{n}\prod_{j=0}^{i-1}(x_i-x_j)$$

因为$x_i\neq x_j(i\neq j)$,所以$V\neq 0$。根据Cramer 法则,该线性方程组有惟一解$a_0,a_1,a_2,\cdots,a_n$,从而$p_n(x)$存在且惟一。

根据定理3.1,由$n+1$个采样点可以惟一地构造出一个次数不高于n的插值多项式$p_n(x)$。在构造该插值多项式时,同样采用基函数线性组合的构造方法。可认为该插值多项式$p_n(x)$由$n+1$个插值基函数线性组合而成,其组合系数就是对应插值节点上的函数值$y_i(i=0,1,2,\cdots,n)$,即:

$$p_n(x)=l_0(x)y_0+l_1(x)y_1+\cdots+l_k(x)y_k+\cdots+l_n(x)y_n$$

或

$$p_n(x)=\sum_{i=0}^{n}l_i(x)y_i \tag{3.8}$$

在插值节点x_i上,该多项式满足插值条件:

$$p_n(x_i)=y_i\quad(i=0,1,2,\cdots,k,\cdots,n)$$

因此,为了使插值条件成立,这$n+1$个插值基函数$l_i(x)(i=0,1,2,\cdots,k,\cdots,n)$在$n+1$个插值节点上必须分别满足:

$$l_i(x_k)=\begin{cases}0, & k\neq i\\ 1, & k=i\end{cases}$$

也就是当插值点为第k个插值节点即$x=x_k$时,在

$$p_n(x_k)=l_0(x_k)y_0+l_1(x_k)y_1+\cdots+l_k(x_k)y_k+\cdots+l_n(x_k)y_n=y_k$$

中,只能是$l_k(x_k)=1$而其他的基函数全部为零$l_i(x_k)=0(i=0,1,\cdots,k-1,k+1,\cdots,n)$。因此,插值基函数$l_i(x)$有一个非零点$x_i$和$n$个零点$x_0,x_1,\cdots,x_{i-1},x_{i+1},\cdots,x_n$,即可设

$$l_i(x)=(x-x_0)(x-x_1)\cdots(x-x_{i-1})\cdot C\cdot(x-x_{i+1})\cdots(x-x_n)=C\prod_{\substack{k=0\\k\neq i}}^{n}(x-x_k)$$

再由$l_i(x_i)=1$,即可确定它的常系数为$C=1\Big/\prod\limits_{\substack{k=0\\k\neq i}}^{n}(x_i-x_k)$,最后得到插值基函数为:

$$l_i(x)=\Big(\prod_{\substack{k=0\\k\neq i}}^{n}(x-x_k)\Big)\Big/\prod_{\substack{k=0\\k\neq i}}^{n}(x_i-x_k)=\prod_{\substack{k=0\\k\neq i}}^{n}\frac{x-x_k}{x_i-x_k}$$

或

$$l_i(x)=\frac{(x-x_0)\cdots(x-x_{i-1})\cdot(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdots(x_i-x_n)}\quad(i=0,1,2,\cdots,n)$$

这样就可得到$n+1$个插值基函数$l_i(x)(i=0,1,2,\cdots,n)$,代入(3.8)式,就得到Lagrange 插值公式:

$$p_n(x)=\sum_{i=0}^{n}l_i(x)\cdot y_i=\sum_{i=0}^{n}\Big(\prod_{\substack{k=0\\k\neq i}}^{n}\frac{x-x_k}{x_i-x_k}\Big)\cdot y_i \tag{3.9}$$

或

$$p_n(x)=\sum_{i=0}^{n}\frac{(x-x_0)\cdots(x-x_{i-1})\cdot(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdots(x_i-x_n)}\cdot y_i$$

Lagrange 插值公式具有以下特点:

(1) 对称性：$p_n(x)$与插值节点的排列顺序无关，只与$(x_i,y_i)(i=0,1,2,\cdots,n)$有关。

(2) $n=1$为线性插值公式，$n=2$为抛物插值公式。

(3) 当插值节点数变化时，基函数需要重新计算。

在附录A中，有计算Lagrange插值的函数Lagrange(x,y,xi)。括号中的参数x,y分别代表已知的采样值；xi是希望通过插值得到函数值点的横坐标数组。Lagrange(x,y,xi)函数的调用格式为：

Yi=Lagrange(x,y,xi)

例3.4 已知采样值为：

x：	1.1	2.3	3.9	5.1
y：	3.887	4.276	4.651	2.117

用MATLAB函数计算在2.101和4.234两处的插值函数值。

解 MATLAB程序如下：

```
clear
x=[1.1,2.3,3.9,5.1];
y=[3.887,4.276,4.651,2.117];
xi=[2.101,4.234];
yi=Lagrange(x,y,xi)
```

运行结果为：

```
yi=
    4.1457 4.3007
```

3.2.4 插值余项

用Lagrange插值公式计算除插值节点以外的某一插值点x处的值，其插值误差为：

$$R_n(x)=f(x)-p_n(x)$$

该误差实际上就是截断误差，称$R_n(x)$为Lagrange插值的插值余项。

定理3.2 设$x_0,x_1,\cdots,x_n$为区间$[a,b]$内的$n+1$个插值节点，而函数$f(x)$在$[a,b]$内有一阶至$n+1$阶导数，且已给定$y_i=f(x_i)(i=0,1,2,\cdots,n)$，则当$x\in[a,b]$时，由Lagrange插值公式(3.9)计算的$p_n(x)$与$f(x)$的误差为：

$$R_n(x)=f(x)-p_n(x)=\frac{f^{(n+1)}(\varphi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i) \tag{3.10}$$

式中φ是依赖x的，它包含在由插值节点$x_0,x_1,\cdots,x_n$和x所界定的(a,b)范围内。

证明 设$x\in[a,b]$为一个插值点，当x取值为$[a,b]$内的插值节点$x_i(i=0,1,2,\cdots,n)$时，(3.10)式显然成立。因此，只需要考虑x取值不是插值节点$x_i(i=0,1,2,\cdots,n)$的情形。

为此，可作辅助函数：

$$\varphi(t)=f(t)-p_n(t)-\frac{\Pi(t)}{\Pi(x)}[f(x)-p_n(x)]$$

这里

$$\Pi(x)=\prod_{i=0}^{n}(x-x_i),\quad \Pi(t)=\prod_{i=0}^{n}(t-x_i)$$

由于$\Pi^{(n+1)}(t)=d^{(n+1)}(t^{(n+1)}+\cdots+x_0x_1\cdots x_n)/dt^{n+1}=(n+1)!$，$d^{(n+1)}p_n(t)/dt^{n+1}=0$，故

$\varphi(t)$为连续函数,且有$n+1$阶导数。当$t=x, x_i(i=0,1,2,\cdots,n)$时,注意到$\Pi(x_k)=\prod_{i=0}^{n}(x_k-x_i)=0$,所以有$\varphi(t)=0$。根据Rolle 定理,有:

当$\varphi(t)$在$[a,b]$内有$n+2$个互异零点时,$\varphi^{(1)}(t)$在$[a,b]$内至少有$n+1$个互异零点;

当$\varphi^{(1)}(t)$在$[a,b]$内有$n+1$个互异零点时,$\varphi^{(2)}(t)$在$[a,b]$内至少有n个互异零点;

……

当$\varphi^{(n)}(t)$在$[a,b]$内有2个互异零点时,$\varphi^{(n+1)}(t)$在$[a,b]$内至少有1个零点。

$\varphi^{(n+1)}(t)$在$[a,b]$内至少有1个零点,记为$\varphi^{(n+1)}(\xi)=0$。

对$\varphi(t)$求$n+1$阶导数,并令$t=\xi$,得:

$$\varphi^{(n+1)}(\xi)=f^{(n+1)}(\xi)-\frac{(n+1)!}{\Pi(x)}[f(x)-p_n(x)]=0$$

故有

$$f(x)-p_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\Pi(x)$$

插值余项公式(3.10)给出了用$p_n(x)$来近似代替$f(x)$的截断误差$R_n(x)$。在实际应用中,对于指定的精度ϵ,要求插值的$R_n(x)\leqslant\epsilon$。

例3.5　已知$\sin30°=1/2$, $\sin45°=\sqrt{2}/2$, $\sin60°=\sqrt{3}/2$,求近似表示$\sin x$的一、二次插值多项式和$\sin50°$的值,并估计误差。

解　(1) 若取30°和45°为插值节点,则一次插值多项式为:

$$p_1(x)=\frac{1}{2}+\frac{\sqrt{2}/2-1/2}{45-30}(x-30)=\frac{1}{2}+\frac{\sqrt{2}-1}{30}(x-30)$$

所以

$$\sin50°\cong p_1(50)=0.77614$$

误差

$$R_1(50)=\frac{f''(\varphi)}{2}(x-x_0)(x-x_1)=\frac{1}{2}(-\sin\varphi)\times\left(\frac{\pi}{180}\right)^2(50-30)(50-45)$$

$$=-50\sin\varphi\left(\frac{\pi}{180}\right)^2=-0.01167\sin\varphi$$

φ在30°~60°之间,故$1/2<\sin\varphi<\sqrt{3}/2$,因此有:

$$-0.01319<R_1(50)<-0.00762$$

(2) 若取45°和60°为插值节点,则一次插值多项式为:

$$p_1(x)=\frac{\sqrt{2}}{2}+\frac{\sqrt{3}/2-\sqrt{2}/2}{60-45}(x-45)=\frac{\sqrt{2}}{2}+\frac{\sqrt{3}-\sqrt{2}}{30}(x-45)$$

所以

$$\sin50°\cong p_1(50)=0.76008$$

误差

$$R_1(50)=\frac{f''(\varphi)}{2}(x-x_0)(x-x_1)=\frac{1}{2}(-\sin\varphi)\times\left(\frac{\pi}{180}\right)^2(50-45)(50-60)$$

$$=25\sin\varphi\left(\frac{\pi}{180}\right)^2=0.005835\sin\varphi$$

φ在30°~60°之间,故$1/2<\sin\varphi<\sqrt{3}/2$,因此有:

$$0.00381<R_1(50)<0.006595$$

(3) 若取30°,45°和60°为插值节点,则二次插值多项式为:

$$p_2(x)=\frac{(x-45)(x-60)}{(30-45)(30-60)}\left(\frac{1}{2}\right)+\frac{(x-30)(x-60)}{(45-30)(45-60)}\left(\frac{\sqrt{2}}{2}\right)+\frac{(x-30)(x-45)}{(60-30)(60-45)}\left(\frac{\sqrt{3}}{2}\right)$$

所以

$$\sin50°\cong p_2(50)=0.76543$$

$$R_2(50)=\frac{1}{3!}(-\cos\varphi)\left(\frac{\pi}{180}\right)^3(50-30)(50-45)(50-60)=0.00068\cos\varphi$$

φ在$30°\sim60°$之间，故$1/2<\cos\varphi<\sqrt{3}/2$，因此有：

$$0.00044<R_2(50)<0.00077$$

从本例中可知，用$x_0=45$和$x_1=60$两点作插值要比用$x_0=30$和$x_1=45$两点作插值精确，这是因为插值点$x=50$在区间[45,60]内部。这种插值称为内插，否则称为外插。一般地，内插要比外插精度高，二次插值要比一次插值精度高。

下面根据插值余项定理推导线性插值的截断误差：

$$R(x)=\frac{f^{(2)}(\xi)}{2}\Pi(x)=\frac{f^{(2)}(\xi)}{2}(x-x_0)(x-x_1)$$

可以证明，函数$\Pi(x)=(x-x_0)(x-x_1)$有极小值$-(x_1-x_0)^2/4$，故

$$|R(x)|=\left|\frac{f^{(2)}(\xi)}{2}(x-x_0)(x-x_1)\right|\leqslant\frac{|f^{(2)}(\xi)|}{8}(x_0-x_1)^2 \tag{3.11}$$

可见，线性插值的误差与函数的二阶导数和插值区间的大小有关。

根据(3.10)式计算插值余项，前提条件是必须已知函数$f(x)$在$[a,b]$内的$n+1$阶导数。而实际情况却恰恰相反，根本不能确定函数$f(x)$及其$n+1$阶导数，仅仅知道离散的插值节点和对应的函数值，即$(x_i,y_i)(i=0,1,2,\cdots,n)$。那么，怎样保证插值的计算精度呢？为此，可以采取下面的方法：

设$L_n(x)$为$f(x)$关于n个插值节点$(x_0,x_1,\cdots,x_n)$的插值多项式，为了估计误差$f(x)-L_n(x)$，另取一个插值节点x_{n+1}，设$\overline{L}_n(x)$为$f(x)$关于n个插值节点$(x_1,\cdots,x_n,x_{n+1})$的插值多项式。根据插值余项定理，有：

$$f(x)-L_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)$$

$$f(x)-\overline{L}_n(x)=\frac{f^{(n+1)}(\eta)}{(n+1)!}\prod_{i=1}^{n+1}(x-x_i)$$

故

$$\frac{f(x)-L_n(x)}{f(x)-\overline{L}_n(x)}=\frac{f^{(n+1)}(\xi)}{f^{(n+1)}(\eta)}\cdot\frac{x-x_0}{x-x_{n+1}}$$

若$f^{(n+1)}(x)$在插值区间上变化不大，则$f^{(n+1)}(\xi)\approx f^{(n+1)}(\eta)$，有：

$$f(x)-L_n(x)\approx\frac{x-x_0}{x_0-x_{n+1}}(L_n(x)-\overline{L}_n(x))$$

只要$L_n(x)-\overline{L}_n(x)$足够小，就能保证$f(x)-L_n(x)<\varepsilon$。这种利用相关两次的插值计算结果来估计误差大小的方法，称为事后误差估计方法。

3.2.5　高次插值的Runge(龙格)现象

在区间$[a,b]$上，运用Lagrange插值公式进行插值，插值节点数越多，插值多项式的次数越高，插值函数和被插值函数重合的点(在插值节点上)也越多，插值计算结果的精度是否也越高呢？下面就来讨论这个问题。

定义3.2　设有$n+1$个插值节点的插值多项式为$p_n(x)$，如果对于任意的$\varepsilon>0$，存在正整数N，当$n>N$时，对被插值的函数$f(x)$及所有$x\in[a,b]$，有

$$f(x)-p_n(x)<\varepsilon$$

成立，则称$p_n(x)$一致收敛于$f(x)$。

考察[$-1,1$]区间内等节距点的$f(x)$的Lagrange插值。对于像$\sin x$和e^x这样性态较好的函数,其所有导数有同样的常数界M,误差$R_n(x)=f(x)-p_n(x)$随着n的增加而趋向零,因此增加插值节点数有利于提高插值结果的精度。但在一般情况下,答案是否定的。例如,对于函数$f(x)=1/(1+x^2)$,当$n\to\infty$时,$R_n(x)$是增加的。

例3.6 给定$f(x)=\frac{1}{1+x^2}(x\in[-5,5])$,取等距节点$x_i=-5+i(i=0,1,\cdots,10)$,试建立插值多项式$P_{10}(x)$,并作出其函数图像,观察$P_{10}(x)$对$f(x)$的逼近效果。

解 构造Lagrange插值多项式为:

$$P_{10}(x)=\sum_{i=0}^{10}l_i(x)f(x_i)$$

其中,插值基函数为:

$$l_i(x)=\frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_{10})}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_{10})}$$

画出的图形如图3.1所示。作为对比,图中还画出了由7个等距构成的插值多项式$P_6(x)$的图形。

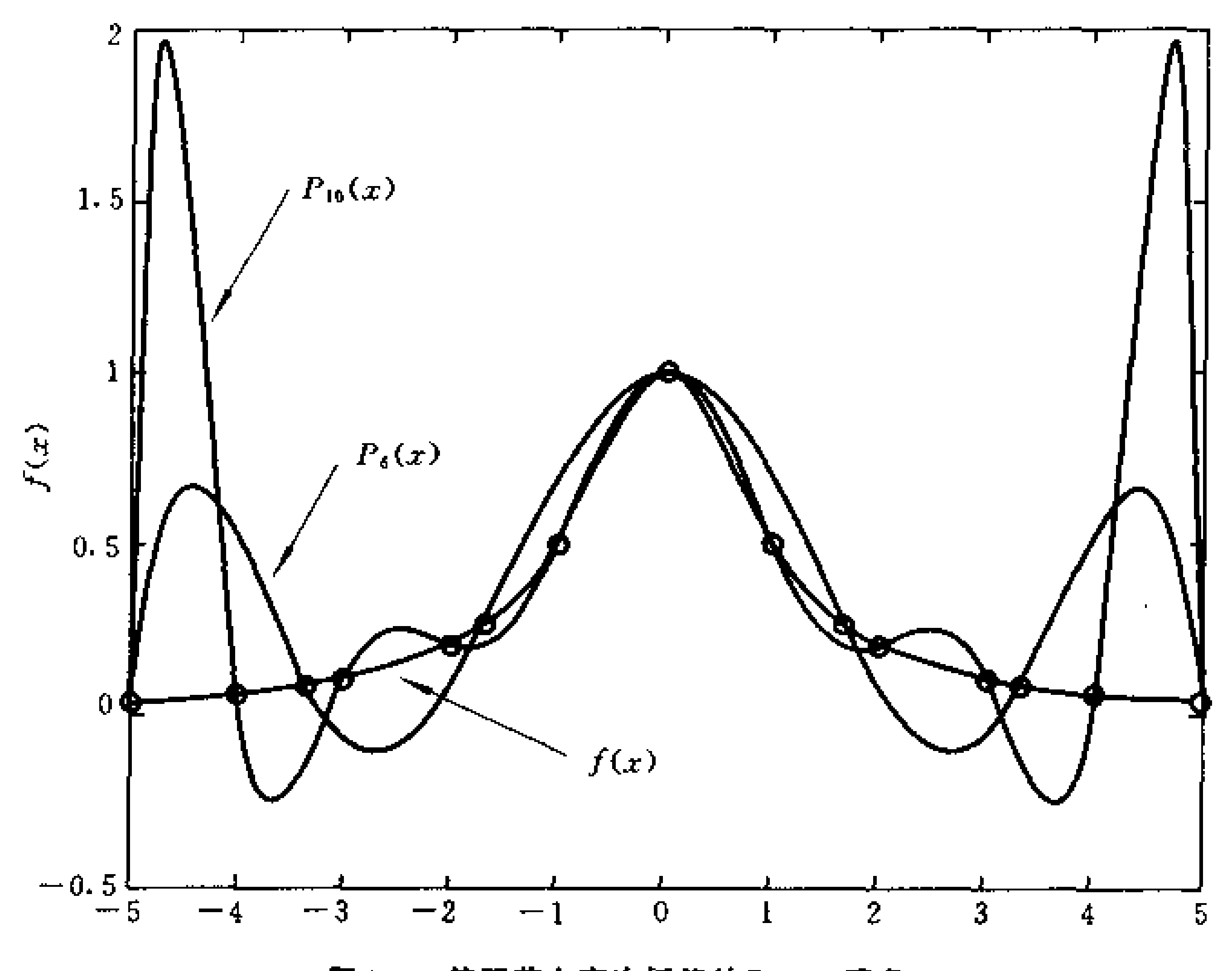


图3.1 等距节点高次插值的Runge现象

从图中可知,当插值点取在[$-2,2$]范围内时,$P_{10}(x)$可以较好地逼近$f(x)$;但在其他插值点处,$P_{10}(x)$与$f(x)$的差异较大,尤其在区间端点的附近,发生了较大的振荡。若节点数增加,则振荡加剧。该问题的出现是由于插值节点是等距造成的。这种对于等距插值节点进行高次插值时所发生的不收敛现象称为Runge现象。从图中还可以看出,对于$f(x)=1/(1+x^2)$这种性态不好的函数,发现使用低次插值如$P_6(x)$效果不理想时,如果企图通过增加插值节点数来提高插值精度,则很容易出现Runge现象,效果反而不如低次插值的效果好。

一般地,为减小Lagrange插值的截断误差,提高插值精度,可以采用如下措施:

(1) 在插值区间内,只能在一定范围($n\leqslant 7$)内依靠增加插值节点的方法提高插值精度,并应该尽量避免使用高次插值,以防止出现Runge现象。

(2) 修改插值条件。如要求插值函数和被插值函数在某些节点具有相同导数,则应采用

Hermite(埃尔米特)插值等。

(3) 减小插值区间或将插值区间分成若干小段，并在每一小段上使用低次插值，即采用分段插值和样条插值。

3.3 逐次插值法与分段插值法

3.3.1 Aitken(埃特金)逐次线性插值法

判断离散数据$(x_i,y_i)(i=0,1,2,\cdots,n)$的插值精度，既可以采用事后误差估计的方法，也可以在插值点x的附近选取部分数据进行插值，然后再增加一些插值节点进行插值。若两次的插值结果之差小于规定的误差，则可认为插值精度符合要求而停止继续插值。这种在插值计算精度不够时增加节点(插值多项式的次数一般不宜超过6～8次)以提高插值精度方法就是所谓的逐次插值法。在上述情况下，运用Lagrange插值方法存在一个明显的缺点，就是当插值节点变化和增加时，Lagrange插值公式中的所有基函数都得重新计算，即计算量大。由于插值节点发生变化和增加只是个别的节点，因此能否只对发生变化和增加的节点进行计算以减小计算量呢？当然，这个问题虽然对使用现代计算机进行计算显得并不重要，但对人工计算就是重要问题。Aitken逐次插值法就是一种可以灵活地增加插值节点数，在前面计算结果的基础上继续进行计算而不必重新开始计算的方法。可见，Aitken逐次插值法具有承袭性的特点。

为方便讨论，可首先约定表示插值结果的符号。即：设在插值区间$[a,b]$上，有$n+1$个顺序排列的插值节点$x_0,\cdots,x_k,\cdots,x_n$，插值点为$x$。由前$k$个顺序排列的插值节点$x_0,x_1,\cdots,x_{k-1}$构成的插值函数是$x$的$k-1$次多项式，可以用$f(x_0,\cdots,x_{k-2};x_{k-1})$表示，简记为$f_{k-1}(x_{k-1})$。在上述$k$个插值节点$x_0,x_1,\cdots,x_{k-1}$的后面，再顺序增加一个新插值节点$x_i(i\geqslant k)$，进行$k$次插值。其插值函数是$x$的$k$次多项式，用$f(x_0,\cdots,x_{k-1};x_i)$表示，简记为$f_k(x_i)$，其中$k$表示插值次数，$x_i$为新增加的插值节点。在简记符号$f_k(x_i)$中，$k$个顺序排列插值节点$x_0,x_1,\cdots,x_{k-1}$中的最后一个节点$x_{k-1}$，由$f_k(x_i)$下标$k$隐含地给出。

1. 一次插值(线性插值)

首先给出一个固定插值节点x_0及其函数值$f(x_0)$，再新增加一个节点$x_i(i\geqslant 1)$(自然同时也给出其函数值$f(x_i)$)，用这两个插值节点进行线性插值，其结果为：

$$f_1(x_i)=\frac{x-x_i}{x_0-x_i}f(x_0)+\frac{x-x_0}{x_i-x_0}f(x_i)\qquad(i\geqslant 1)$$

若取$i=1$，则表示以x_0,x_1为节点进行一次插值，结果为：

$$f_1(x_1)=\frac{x-x_1}{x_0-x_1}f(x_0)+\frac{x-x_0}{x_1-x_0}f(x_1)$$

若取$i=2$，则表示以x_0和x_2为节点进行一次插值，结果为：

$$f_1(x_2)=\frac{x-x_2}{x_0-x_2}f(x_0)+\frac{x-x_0}{x_2-x_0}f(x_2)$$

由此可得出如下结论：$f_1(x_i)$表示取固定节点x_0和变化节点$x_i(i\geqslant 1)$及其相应的$f(x_0)$，$f(x_i)$进行线性插值，得到关于x的1次多项式。

2. 二次插值

顺序给出两个插值节点x_0,x_1，再新增加一个节点$x_i(i\geqslant 2)$，用这3个插值节点进行插值，

其结果为：

$$f_2(x_i)=\frac{(x-x_1)(x-x_i)}{(x_0-x_1)(x_0-x_i)}f(x_0)+\frac{(x-x_0)(x-x_i)}{(x_1-x_0)(x_1-x_i)}f(x_1)+\frac{(x-x_0)(x-x_1)}{(x_i-x_0)(x_i-x_1)}f(x_i),(i\geqslant 2)$$

若取 $i=2$，则表示以 x_0，x_1 和 x_2 为节点进行二次插值，其结果为：

$$f_2(x_2)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}f(x_0)+\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}f(x_1)+\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}f(x_2)$$

若取 $i=3$，则表示以 x_0，x_1 和 x_3 为节点进行二次插值，其结果为：

$$f_2(x_3)=\frac{(x-x_1)(x-x_3)}{(x_0-x_1)(x_0-x_3)}f(x_0)+\frac{(x-x_0)(x-x_3)}{(x_1-x_0)(x_1-x_3)}f(x_1)+\frac{(x-x_0)(x-x_3)}{(x_3-x_0)(x_2-x_1)}f(x_3)$$

整理 $f_2(x_2)$，得：

$$f_2(x_2)=\frac{x-x_2}{x_1-x_2}f_1(x_1)+\frac{x-x_1}{x_2-x_1}f_1(x_2)$$

同理

$$f_2(x_i)=\frac{x-x_i}{x_1-x_i}f_1(x_1)+\frac{x-x_1}{x_i-x_1}f_1(x_i)$$

由此可得出如下结论：$f_2(x_i)$ 表示取固定节点 x_1 和变化节点 $x_i(i\geqslant 2)$ 及其相应的 $f_1(x_1)$，$f_1(x_i)$ 进行线性插值，得到关于 x 的 2 次多项式。

3. k 次插值

根据以上分析，可以推出如下结论：用两个 $k-1$ 次插值的结果 $f_{k-1}(x_{k-1})$ 和 $f_{k-1}(x_i)$，进行线性插值，即可得到 k 次插值的结果 $f_k(x_i)$。即：

$f_k(x_i)$ 表示取固定节点 x_{k-1} 和变化节点 $x_i(i\geqslant k)$ 及其相应的 $f_{k-1}(x_{k-1})$，$f_{k-1}(x_i)$ 进行线性插值，从而得到关于 x 的 k 次多项式：

$$f_k(x_i)=\frac{x-x_i}{x_{k-1}-x_i}f_{k-1}(x_{k-1})+\frac{x-x_{k-1}}{x_i-x_{k-1}}f_{k-1}(x_i),\quad (i\geqslant k) \tag{3.12}$$

根据(3.12)式，可以计算出当 $k=1,2,\cdots,n(i=k,k+1,\cdots,n)$ 时的 $f_k(x_i)$，由于计算 $f_k(x_i)$ 有很强的规律性，故将其排列成表3.1 所示的格式，该表称为Aitken 插值表。从表中可以看出，$f_k(x_i)$ 是逐列计算出来的。这种逐步提高插值次数以获得更高精度插值结果的插值方法称为 Aitken 逐次插值方法。

表 3.1　Aitken 插值表($k\leqslant 4$ 部分)

x_i	$f(x_i)$	$f_1(x_0;x_i)$	$f_2(x_1;x_i)$	$f_3(x_2;x_i)$	$f_4(x_3;x_i)$
x_0	$f(x_0)$				
x_1	$f(x_1)$	$f_1(x_1)$			
x_2	$f(x_1)$	$f_1(x_2)$	$f_2(x_2)$		
x_3	$f(x_3)$	$f_1(x_3)$	$f_2(x_3)$	$f_3(x_3)$	
x_4	$f(x_4)$	$f_1(x_4)$	$f_2(x_4)$	$f_3(x_4)$	$f_4(x_4)$
⋮	⋮	⋮	⋮	⋮	⋮

注：表中 $f_k(x_{k-1};x_i)$ 表示强调以 x_{k-1} 为固定插值节点、x_i 为变化节点进行线性插值。

归纳起来，Aitken 逐次插值法的特点是：

(1) 将一个高次插值过程归结为线性插值的多次重复。

(2) 插值表中的每个数据均为插值结果。从这些数据的一致程度可判断插值结果的精度，如果未达到精度要求，则再增加一个节点进行插值，直至满意为止。

4. 编程计算格式

应用计算机编程计算，Aitken 逐次插值公式(3.12)可改写为：

$$f_k(x_i)=f_{k-1}(x_{k-1})+\frac{x-x_{k-1}}{x_i-x_{k-1}}[f_{k-1}(x_i)-f_{k-1}(x_{k-1})]$$

进而写成动态形式：

$$y_i=y_{k-1}+\frac{x-x_{k-1}}{x_i-x_{k-1}}(y_i-y_{k-1}),\quad (i=k,k+1,\cdots,n) \tag{3.13}$$

(3.13)式就是计算机编程用的格式。

3.3.2 分段插值法

由前面的讨论可知，用多项式作为插值函数来逼近某一函数 $f(x)$ 是最简单易行的一种插值方法，但是插值多项式的次数是随着插值节点的数目而增加的，且次数高的插值多项式往往插值效果并不理想，会出现所谓的Runge 现象，即在插值函数 $p_n(x)$ 的两端会发生激烈地震荡(不稳定)。为此，在实际应用中常采用分段插值方法。

所谓分段插值法就是将被插值函数逐段多项式化，构造一个分段多项式作为插值函数。

分段插值的做法是：首先，将插值区间划分为若干小段，在每一小段上使用低阶插值；然后，将各小段上的插值多项式拼接在一起作为整个区间上的插值函数。如果使用的低阶插值为线性插值(两点插值)，则将拼接成一条折线，用它来逼近函数 $f(x)$。

应用低阶插值的关键在于恰当地选择插值节点。由插值余项公式(3.10)可知，所选节点 x_i 离插值点 x 越近则误差越小。

1. 分段线性插值

将插值区间 $[a,b]$ 分成

$$a=x_0,x_1,x_2,\cdots,x_n=b$$

n 个小段，在每一个小段 $[x_{i-1},x_i]$ $(i=1,2,\cdots,n)$ 上，其分段线性插值的公式为：

$$s(x)=y_i+\frac{y_i-y_{i-1}}{x_i-x_{i-1}}(x-x_i) \tag{3.14}$$

根据

$$i=\begin{cases}1 & x\leqslant x_0\\ k & x_{k-1}<x\leqslant x_k \text{ 时},(1\leqslant k\leqslant n)\\ n & x>x_n\end{cases}$$

选择插值节点，即当插值节点为 $x_0,x_1,x_2,\cdots,x_{k-1},x_k,\cdots,x_n$ 时，依次从左至右取出各节点。如果插值点 x 不超过节点 x_1(即在 $[x_0,x_1]$ 之间)，则取节点 x_0 和 x_1 进行线性插值，否则，再检查 x 是否超过 x_2，…，依次逐步检查。一旦发现 x 不超过某个节点 x_i，则取它与前面一个节点 x_{i-1} 进行线性插值。如果 x 已超过 x_{n-1}，则不论是否超过 x_n，插值节点均取 x_n 和 x_{n-1}(也就是一律当成是在 $[x_{n-1},x_n]$ 范围内取插值点)。

根据(3.11)式，在小段 $[x_{i-1},x_i]$ 上，分段线性插值的误差是：

$$|R_i(x)|=|f(x)-s(x)|\leqslant\frac{|f^{(2)}(\xi)|}{8}(x_i-x_{i-1})^2,\quad \xi\in[x_{i-1},x_i]$$

可见,当$f^{(2)}(\xi)$有界时,小段$[x_{i-1},x_i]$越小,分段线性插值的误差就越小。用分段线性插值方法提高插值精度是有效的。

2. 分段抛物插值

为了提高插值精度,可以在每一小段取3个节点x_{i-1},x_i和x_{i+1}进行二次插值,从而构成分段抛物插值。其插值公式如下:

$$y=\frac{(x-x_i)(x-x_{i+1})}{(x_{i-1}-x_i)(x_{i-1}-x_{i+1})}\cdot y_{i-1}+\frac{(x-x_{i-1})(x-x_{i+1})}{(x_i-x_{i-1})(x_i-x_{i+1})}\cdot y_i$$

$$+\frac{(x-x_{i-1})(x-x_i)}{(x_{i+1}-x_{i-1})(x_{i+1}-x_i)}\cdot y_{i+1}\tag{3.15}$$

根据

$$i=\begin{cases}1 & x<x_1\\ k-1 & x_{k-1}<x<x_k\text{且}|x-x_{k-1}|\leqslant|x-x_k|,k=2,3,\cdots,n-1\\ k & x_{k-1}<x<x_k\text{且}|x-x_{k-1}|>|x-x_k|,k=2,3,\cdots,n-1\\ n-1 & x>x_{n-1}\end{cases}$$

选择插值节点。即靠近x_0取$i=1$,计算节点为x_0,x_1,x_2;靠近x_{k-1}取$i=k-1$,计算节点为x_{k-2},x_{k-1},x_k;靠近x_k取$i=k$,计算节点为x_{k-1},x_k,x_{k+1};靠近x_n取$i=n-1$,计算节点为x_{n-2},x_{n-1},x_n。

3. 分段插值方法的特点

(1) 分段插值方法算法简单,收敛性可以得到保证,只要节点间距充分小,就能达到任何精度的要求。

(2) 如需修改某个数据,则插值函数仅在相关的某个局部范围内受影响。

(3) 分段抛物插值所拼接成的插值函数曲线不一定光滑。

对例3.6中的函数$f(x)=1/(1+x^2)$,采用分段插值的方法,效果比较好。在图3.2所示的分段插值效果图中,由−5,−4,−3,−2这4个节点构成一个分段插值,由−1,0,1这3个节点构成一个分段插值,其插值效果都比较理想。

3.4 Newton(牛顿)插值法

Aitken逐次插值法虽然具有承袭性的特点,但其插值公式是递推型的,不便于进行理论分析。为此,可以把n次插值多项式改写成升幂的形式:

$$N_n(x)=c_0+c_1(x-x_0)+c_2(x-x_0)(x-x_1)+\cdots+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})\tag{3.16}$$

其中,$c_0,c_1,c_2,\cdots,c_n$为待定系数。根据定理3.1,该多项式是惟一存在的。将插值条件

$$N_n(x_i)=f(x_i)\quad(i=0,1,2,\cdots,n)$$

代入(3.16)式中,即可以惟一确定出系数$c_0,c_1,c_2,\cdots,c_n$,从而得到$N_n(x)$的升幂形式。为了方便计算,在具体计算这些系数之前,先引入差商的概念。

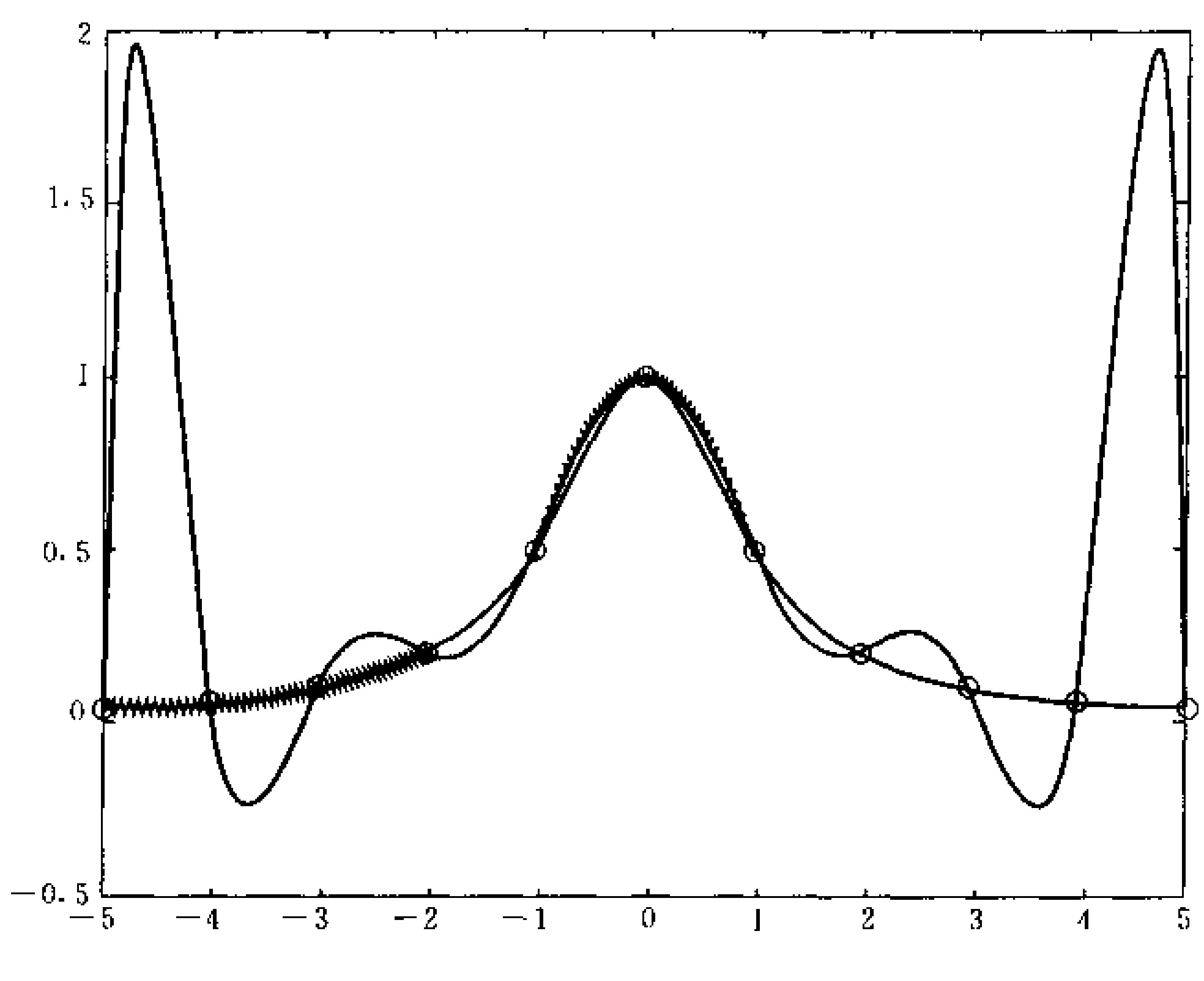


图3.2 分段插值的效果

1. 差商的定义与性质

定义3.3 已知顺序排列的节点$x_0,x_1,x_2,x_3,\cdots,x_{k-1},x_k,\cdots,x_n$所对应的函数值为$f(x_0)$，$f(x_1),f(x_2),\cdots,f(x_{k-1}),f(x_k),\cdots,f(x_n)$，定义

$$f[x_0,x_k]=\frac{f(x_0)-f(x_k)}{x_0-x_k}\quad(1\leqslant k\leqslant n)$$

为函数$f(x)$在点x_0,x_k处的一阶差商；定义

$$f[x_0;x_1,x_k]=\frac{f[x_0,x_1]-f[x_0,x_k]}{x_1-x_k}\quad(2\leqslant k\leqslant n)$$

为函数$f(x)$在点x_0,x_1,x_k处的二阶差商；定义

$$f[x_0,x_1;x_2,x_k]=\frac{f[x_0;x_1,x_2]-f[x_0;x_1,x_k]}{x_2-x_k}\quad(3\leqslant k\leqslant n)$$

为函数$f(x)$在点x_0,x_1,x_2,x_k处的三阶差商；依此类推，定义

$$f[x_0,x_1,\cdots,x_{k-2};x_{k-1},x_k]=\frac{f[x_0,x_1,\cdots;x_{k-2},x_{k-1}]-f[x_0,x_1,\cdots;x_{k-2},x_k]}{x_{k-1}-x_k}\quad(k\leqslant n)\tag{3.17}$$

为函数$f(x)$在点$x_0,x_1,x_2,x_3,\cdots,x_{k-1},x_k$处的$k$阶差商。

k阶差商还有另外一种定义方法，即

$$f[x_0,x_1,\cdots,x_{k-1},x_k]=\frac{f[x_0,x_1,\cdots,x_{k-1}]-f[x_1,\cdots,x_{k-1},x_k]}{x_0-x_k}\tag{3.18}$$

故称(3.17)式为第一种格式的差商，(3.18)式为第二种格式的差商。两者具有完全相同的性质。

差商有如下性质：

(1) k 阶差商 $f[x_0,x_1,\cdots,x_k]$ 是函数值 $f(x_0),f(x_1),\cdots,f(x_n)$ 的线性组合。即

$$f[x_0,x_1,\cdots,x_k]=\sum_{i=0}^{k}\frac{f(x_i)}{(x_i-x_0)(x_i-x_1)\cdots(x_i-x_{i-1})\cdot(x_i-x_{i+1})\cdots(x_i-x_k)}$$

(2) 差商的值与节点的排列顺序无关。即

$$f[x_0,x_1,\cdots,x_k]=f[x_1,x_0,\cdots,x_k]=\cdots=f[x_k,\cdots,x_1,x_0]$$

(3) 若 $f[x,x_0,x_1,\cdots,x_k]$ 是 x 的 m 次多项式,则 $f[x,x_0,x_1,\cdots,x_k,x_{k+1}]$ 是 x 的 $m-1$ 次多项式。

2. Newton 插值公式

为了求出Newton 插值多项式

$$\begin{aligned}N_n(x)=c_0&\\&+c_1(x-x_0)\\&+c_2(x-x_0)(x-x_1)\\&+c_3(x-x_0)(x-x_1)(x-x_2)\\&+\cdots\\&+c_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})\end{aligned}$$

的系数,将 $N_n(x_i)=f(x_i)(i=0,1,2,\cdots,n)$ 按插值节点的顺序逐个代入上式,即:

由 $N_n(x_0)=c_0=f(x_0)$,得:$c_0=f(x_0)$

由 $N_n(x_1)=f(x_0)+c_1(x_1-x_0)=f(x_1)$,得:

$$c_1=\frac{f(x_1)-f(x_0)}{(x_1-x_0)}=\frac{f(x_0)-f(x_1)}{(x_0-x_1)}=f[x_0,x_1]$$

由 $N_n(x_2)=f(x_0)+f[x_0,x_1](x_2-x_0)+c_2(x_2-x_0)(x_2-x_1)=f(x_2)$,得:

$$c_2=\frac{f(x_2)-f(x_0)-f[x_0,x_1](x_2-x_0)}{(x_2-x_0)(x_2-x_1)}=\frac{\dfrac{f(x_2)-f(x_0)}{(x_2-x_0)}-f[x_0,x_1]}{(x_2-x_1)}$$

$$=\frac{f[x_0,x_1]-\dfrac{f(x_0)-f(x_2)}{(x_0-x_2)}}{(x_1-x_2)}=\frac{f[x_0,x_1]-f[x_0,x_2]}{(x_1-x_2)}=f[x_0,x_1,x_2]$$

由 $N_n(x_3)=f(x_0)+f[x_0,x_1](x_3-x_0)+f[x_0;x_1,x_2](x_3-x_0)(x_3-x_1)+c_3(x_3-x_0)(x_3-x_1)(x_3-x_2)=f(x_3)$,得:

$$c_3=\frac{f(x_3)-f(x_0)-f[x_0,x_1](x_3-x_0)-f[x_0;x_1,x_2](x_3-x_0)(x_3-x_1)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)}$$

$$=\frac{f[x_0,x_3]-f[x_0,x_1]-f[x_0;x_1,x_2](x_3-x_1)}{(x_3-x_1)(x_3-x_2)}$$

$$=\frac{f[x_0;x_1,x_3]-f[x_0;x_1,x_2]}{(x_3-x_2)}=f[x_0,x_1;x_2,x_3]$$

依此类推,可由差商的定义和数学归纳法得出全部系数,即:

$$c_k=f[x_0,x_1,\cdots,x_k],\quad(k=0,1,2,\cdots,n)$$

这样就得到了Newton 插值公式:

$$\begin{aligned}N_n(x)=f(x_0)&+f[x_0,x_1](x-x_0)+f[x_0;x_1,x_2](x-x_0)(x-x_1)\\&+\cdots+f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})\end{aligned}\tag{3.19}$$

用Newton 插值法构造插值多项式时,只需计算各个节点间的各阶差商即可。由于各阶差

商的计算具有递推规律,因此,将各阶差商排列在一起,组成差商表。第一种、第二种格式的差商表分别见表3.2和表3.3。

表3.2 第一种格式的差商表(四阶差商以下部分)

x	$f(x)$	一阶差商	二阶差商	三阶差商	四阶差商
x_0	$f(x_0)$				
x_1	$f(x_1)$	$f[x_0,x_1]$			
x_2	$f(x_2)$	$f[x_0,x_2]$	$f[x_0;x_1,x_2]$		
x_3	$f(x_3)$	$f[x_0,x_3]$	$f[x_0;x_1,x_3]$	$f[x_0,x_1;x_2,x_3]$	
x_4	$f(x_4)$	$f[x_0,x_4]$	$f[x_0;x_1,x_4]$	$f[x_0;x_1;x_2,x_4]$	$f[x_0,x_1,x_2;x_3,x_4]$
…	…	…	…	…	…

表3.3 第二种格式的差商表(四阶差商以下部分)

x	$f(x)$	一阶差商	二阶差商	三阶差商	四阶差商
x_0	$f(x_0)$				
		$f[x_0,x_1]$			
x_1	$f(x_1)$		$f[x_0,x_1,x_2]$		
		$f[x_1,x_2]$		$f[x_0,x_1,x_2,x_3]$	
x_2	$f(x_2)$		$f[x_1,x_2,x_3]$		$f[x_0,x_1,x_2,x_3,x_4]$
		$f[x_2,x_3]$		$f[x_1,x_2,x_3,x_4]$	
x_3	$f(x_3)$		$f[x_2,x_3,x_4]$		…
		$f[x_3,x_4]$		…	
x_4	$f(x_4)$		…		
…	…	…			

3. 插值余项

如果把插值点x看成是$[a,b]$上的一固定点,由一阶差商定义,有:

$$f(x)=f(x_0)+f[x_0,x](x-x_0)$$

由二阶差商定义,有:

$$f[x_0,x]=f[x_0,x_1]+f[x_0;x_1,x](x-x_1)$$

由三阶差商定义,有:

$$f[x_0;x_1,x]=f[x_0;x_1,x_2]+f[x_0,x_1;x_2,x](x-x_2)$$

依此类推,有:

$$f[x_0,x_1,\cdots;x_{n-1},x]=f[x_0,x_1,\cdots,x_n]+f[x_0,x_1,\cdots;x_n,x](x-x_n)$$

将后一等式逐项代入前一等式,得:

$$\begin{aligned}f(x)&=f(x_0)+f[x_0,x_1](x-x_0)+\cdots+f[x_0,x_1,\cdots,x_n](x-x_0)(x-x_1)\cdots(x-x_{n-1})\\&\quad+f[x_0,x_1,\cdots;x_n,x](x-x_0)(x-x_1)\cdots(x-x_n)\\&=N_n(x)+R_n(x)\end{aligned}$$

其中

$$R_n(x)=f[x_0,x_1,\cdots,x_n,x](x-x_0)(x-x_1)\cdots(x-x_n) \tag{3.20}$$

称为Newton插值的余项。

由于对相同插值节点,插值多项式是惟一的,所以,Newton插值多项式与Lagrange插值

多项式是等价的。同样,两者的余项也是等价的。因此,当 $f^{(n+1)}(x)$ 存在时,有:

$$f[x_0,x_1,\cdots,x_n,x]=\frac{f^{(n+1)}(\xi)}{(n+1)!}$$

例 3.7　依据下列函数值建立不超过 3 次的 Lagrange 插值多项式及 Newton 插值多项式,并验证插值多项式的惟一性。

x	0	1	2	4
$f(x)$	1	9	23	3

解　(1) 对于 Lagrange 插值多项式,其插值基函数为:

$$l_0(x)=\frac{(x-1)(x-2)(x-4)}{(0-1)(0-2)(0-4)}=-\frac{1}{8}x^3+\frac{7}{8}x^2-\frac{7}{4}x+1$$

$$l_1(x)=\frac{(x-0)(x-2)(x-4)}{(1-0)(1-2)(1-4)}=\frac{1}{3}x^3-2x^2+\frac{8}{3}x$$

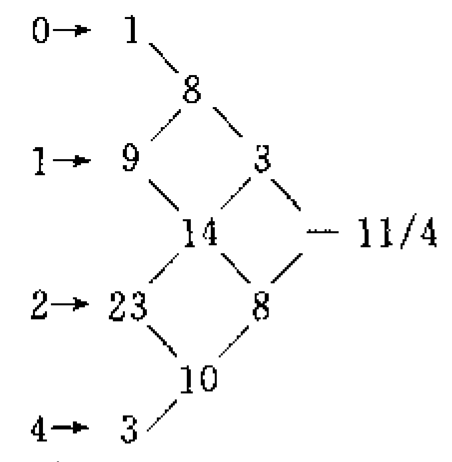
$$l_2(x)=\frac{(x-0)(x-1)(x-4)}{(2-0)(2-1)(2-4)}=-\frac{1}{4}x^3+\frac{5}{4}x^2-x$$

$$l_3(x)=\frac{(x-0)(x-1)(x-2)}{(4-0)(4-1)(4-2)}=\frac{1}{24}x^3-\frac{1}{8}x^2+\frac{1}{12}x$$

于是,Lagrange 插值多项式为:

$$L_3(x)=\sum_{i=0}^{3}f(x_0)l_i(x)=l_0(x)+9l_1(x)+23l_2(x)+3l_3(x)=-\frac{11}{4}x^3+\frac{45}{4}x^2-\frac{1}{2}x+1$$

(2) 对于 Newton 插值多项式,其差商表为:



牛顿插值多项式为:

$$N_3(x)=1+8(x-0)+3(x-0)(x-1)-\frac{11}{4}(x-0)(x-1)(x-2)=-\frac{11}{4}x^3+\frac{45}{4}x^2-\frac{1}{2}x+1$$

因此,用 Lagrange 插值法和 Newton 插值法对同一插值问题的结果是一致的。

思考题:如果将插值节点的顺序改变为 2,4,1,0,其 Newton 插值的结果还一样吗?为什么?

例 3.8　给定下列插值数据,求 4 次 Newton 插值多项式,并写出其插值余项。

x	1	2	4	6	7
$f(x)$	4	1	0	1	1

解　按第一种格式计算差商:

x	$f(x)$	一阶差商	二阶差商	三阶差商	四阶差商
1	4				
2	1	−3			
4	0	−4/3	5/6		
6	1	−3/5	3/5	−7/60	
7	1	−1/2	1/2	−1/9	1/180

按第二种格式计算差商：

x	$f(x)$	一阶差商	二阶差商	三阶差商	四阶差商
1→	4				
		-3			
2→	0		5/6		
		$-1/2$		$-7/60$	
4→	1		1/4		1/180
		1/2		$-1/12$	
6→	1		$-1/6$		
		0			
7→	1				

由差商表可得4次Newton插值多项式为：

$$N_4(x)=4-3(x-1)-\frac{5}{6}(x-1)(x-2)-\frac{7}{60}(x-1)(x-2)(x-4)$$

$$+\frac{1}{180}(x-1)(x-2)(x-4)(x-6)$$

插值余项为：

$$f(x)-N_4(x)=\frac{f^{(5)}(\xi)}{5!}(x-1)(x-2)(x-4)(x-6)(x-7)$$

$$\xi\in(\min(x,1),\max(x,7))$$

3.5 Hermite(埃尔米特)插值法

前面所介绍的插值，只要求插值多项式$p_n(x)$与被插值函数$f(x)$在插值节点处的函数值相等即可，即$p_n(x_i)=f(x_i)(i=0,1,\cdots,n)$。但这种插值不能完全反映出被插值函数的性态。在许多实际问题中，不仅要求插值函数与被插值函数在节点处的函数值相同，而且还要求插值函数与被插值函数在某些节点处的导数值、甚至高阶导数值也相同。按照这种插值条件所进行的插值称为Hermite插值。其中，最常见的是要求一阶导数值相同的插值。下面就来导出其插值多项式。

3.5.1 Hermite插值多项式

假设已知函数$y=f(x)$在插值节点$x_i(i=0,1,\cdots,n)$处的函数值为$y_i=f(x_i)$，一阶导数值为$m_i=f'(x_i)(i=0,1,2,\cdots,n)$，则Hermite插值多项式$H(x)$满足：

$$\begin{cases}H(x_i)=y_i,\\ H'(x_i)=m_i,\end{cases}\quad i=0,1,\cdots,n \tag{3.21}$$

其几何意义就是，要求$y=H(x)$的图形与$y=f(x)$的图形在这$n+1$个点处相切，因此$x_0,x_1,\cdots,x_n$称为二重节点。在(3.21)式中有$2(n+1)$个条件，因此可以确定一个次数不超过$2n+1$次的插值多项式。仿照求Lagrange插值多项式的方法，设

$$H(x)=\alpha_0(x)y_0+\beta_0(x)m_0+\alpha_1(x)y_1+\beta_1(x)m_1+\cdots+\alpha_n(x)y_n+\beta_n(x)m_n$$

或
$$H(x)=\sum_{i=0}^{n}(\alpha_i(x)y_i+\beta_i(x)m_i) \tag{3.22}$$

根据插值条件，$\alpha_i(x)$与$\beta_i(x)$必须分别满足：

$$\begin{cases}\alpha_i(x_k)=\begin{cases}0, k\neq i\\ 1, k=i\end{cases} & i,k=0,1,\cdots,n\\ \alpha_i'(x_k)=0\end{cases} \tag{3.23}$$

和
$$\begin{cases}\beta_i(x_k)=0\\ \beta_i'(x_k)=\begin{cases}0, k\neq i & i,k=0,1,\cdots,n\\ 1, k=i\end{cases}\end{cases} \tag{3.24}$$

这样确定的$\alpha_i(x)$与$\beta_i(x)$称为Hermite插值基函数。显然，基函数$\alpha_i(x)$与$\beta_i(x)$一旦确定下来，Hermite插值多项式$H(x)$就确定出来了。下面导出$\alpha_i(x)$与$\beta_i(x)$的具体形式：

(1) 先确定$\beta_i(x)$。

由
$$\beta_i(x_k)=\beta_i'(x_k)=0 \quad (k\neq i)$$
知$\beta_i(x_i)$以$x_k(k\neq i)$为二重零点，又由
$$\beta_i(x_i)=0,\quad \beta_i'(x_i)=1\neq 0$$
知$\beta_i(x)$以x_i为一重零点。由于$\beta_i(x)$为次数不超过$2n+1$的多项式，再根据Lagrange基函数$l_i(x)$的定义，可设
$$\beta_i(x)=\bar{\beta}_i\times(x-x_i)\times l_i^2(x) \quad \bar{\beta}_i\text{为常数}$$

由$\beta_i'(x_i)=1$，得$\bar{\beta}_i=1$，故有：
$$\beta_i(x)=(x-x_i)l_i^2(x) \tag{3.25}$$

(2) 再确定$\alpha_i(x)$。

同理，根据$\alpha_i(x)$的定义，$\alpha_i(x)$含有因子$l_i^2(x)$，故可令
$$\alpha_i(x)=(ax+b)\times l_i^2(x)$$
其中，a，b为特定常数。于是由条件
$$\begin{cases}\alpha_i(x_i)=1\\ \alpha_i'(x_i)=0\end{cases}$$
可得出：
$$\begin{cases}ax_i+b=1,\\ a+2l_i'(x_i)=0\end{cases}\Rightarrow\begin{cases}a=-2l_i'(x_i)\\ b=1+2x_il_i'(x_i)\end{cases}$$

用对数法对$l_i(x)=\dfrac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$求导，得$l_i'(x_i)=\sum\limits_{\substack{k=0\\k\neq i}}^{n}\dfrac{1}{x_i-x_k}$，故
$$\alpha_i(x)=\left(1-2(x-x_i)\sum_{\substack{k=0\\k\neq i}}^{n}\frac{1}{x_i-x_k}\right)l_i^2(x) \tag{3.26}$$

将(3.25)式和(3.26)式代入(3.22)式中，即可得到Hermite插值多项式为：
$$H(x)=\sum_{i=0}^{n}\left\{y_i+(x_i-x)\left[\left(2\sum_{\substack{k=0\\k\neq i}}^{n}\frac{1}{x_i-x_k}\right)y_i-m_i\right]\right\}\times l_i^2(x) \tag{3.27}$$

(3) 最后证明Hermite插值多项式的惟一性。

假设另有一个不超过$2n+1$次的多项式$h(x)$并满足Hermite插值条件，设
$$\varphi(x)=H(x)-h(x)$$

则有：

$$\varphi(x_i)=H(x_i)-h(x_i)=f(x_i)-f(x_i)=0,\quad (i=0,1,\cdots,n)$$

$$\varphi'(x_i)=H'(x_i)-h'(x_i)=m_i-m_i=0,\quad (i=0,1,\cdots,n)$$

于是$\varphi(x)$有$2n+2$个零点，但是由于$\varphi(x)$是次数不超过$2n+1$的多项式，所以$\varphi(x)\equiv 0$，从而有：

$$H(x)=h(x)$$

3.5.2 Hermite 插值余项

设$f(x)$在插值区间上有$2n+2$阶导数，令$R(x)=f(x)-H(x)$，由$H(x)$的定义知，$x_i(i=0,1,\cdots,n)$是$R(x)$的二重零点，因此$R(x)$含有$(x-x_i)$因子，故可设

$$R(x)=k(x)\Pi^2(x)=k(x)\cdot[(x-x_0)(x-x_1)\cdots(x-x_n)]^2$$

作辅助函数

$$\varphi(t)=f(t)-H(t)-k(x)\Pi^2(t)$$

则有

$$\varphi(x_i)=0,\quad \varphi'(x_i)=0\quad (i=0,1,\cdots,n)$$

$$\varphi(x)=0$$

所以，$\varphi(t)$至少有$n+1$个二重零点$x_i(i=0,1,\cdots,n)$和一个单零点x。按照Rolle定理，$\varphi'(t)$在$x_0,x_1,\cdots,x_n,x$每相邻两个零点之间各有一个零点，且$x_0,x_1,\cdots,x_n$仍是$\varphi'(t)$的零点。所以，$\varphi'(t)$至少有$2n+2$个零点。同理$\varphi''(t)$在插值区间内至少有$2n+1$个零点，依次类推，$\varphi^{(2n+2)}(t)$在插值区间内至少有一个零点ξ，即：

$$\varphi^{(2n+2)}(\xi)=f^{(2n+2)}(\xi)-k(x)\cdot(2n+2)!=0$$

所以

$$k(x)=\frac{f^{(2n+2)}(\xi)}{(2n+2)!}$$

由此得余项为：

$$R(x)=\frac{f^{(2n+2)}(\xi)}{(2n+2)!}\Pi^2(x)\quad (\text{其中}\xi\text{依赖于变量}x)\tag{3.28}$$

应用最广泛的是三次Hermite插值多项式，即$n=1$时的情况：

$$H_3(x)=\left(1+2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2y_0+\left(1+2\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_0}{x_1-x_0}\right)^2y_1$$
$$+(x-x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2m_0+(x-x_1)\left(\frac{x-x_0}{x_1-x_0}\right)^2m_1\tag{3.29}$$

插值余项为：

$$R_3(x)=f(x)-H_3(x)=\frac{f^{(4)}(\xi)}{4!}(x-x_0)^2(x-x_1)^2\quad \xi\in(x_0,x_1)\tag{3.30}$$

例3.9 设$f(x)=\ln x$，节点$x_0=2.2$，$x_1=2.4$，用$H_3(x)$逼近$f(x)$，计算$f(2.3)$。

解 $m_0=\dfrac{1}{x_0}=\dfrac{1}{2.2}=0.454545$，$m_1=\dfrac{1}{x_1}=\dfrac{1}{2.4}=0.41667$

$$H_3(x)=\left(1+2\frac{x-2.2}{2.4-2.2}\right)\left(\frac{x-2.4}{2.2-2.4}\right)^2\times 0.78846$$
$$+\left(1+2\frac{x-2.4}{2.2-2.4}\right)\left(\frac{x-2.2}{2.4-2.2}\right)^2\times 0.87547$$
$$+(x-2.2)\left(\frac{x-2.4}{2.2-2.4}\right)^2\times 0.454545$$
$$+(x-2.4)\left(\frac{x-2.2}{2.4-2.2}\right)^2\times 0.416667$$

$$f(2.3)=\ln(2.3)\approx H_3(2.3)=0.83291$$

$$|R_3(2.3)|=\left|\frac{1}{4!}\left(-\frac{6}{\xi^4}\right)(2.3-2.2)^2(2.3-2.4)^2\right|$$

$$\leqslant\frac{1}{4!}\left(\frac{6}{2.2^4}\right)(2.3-2.2)^2(2.3-2.4)^2=1.067\times10^{-6}$$

3.6 曲线拟合方法

插值方法要求插值曲线严格通过所给的每一个数据点(样点),数据本身存在的不可避免的误差会反映到插值结果中。如果数据误差很大,用插值方法显然就不恰当,就应该采用曲线拟合方法。

和插值方法相比,曲线拟合方法要求给出多得多的数据点,但所给的数据不一定可靠(即有误差),个别数据误差甚至很大。因此,曲线拟合方法是研究从给出的一大堆看上去似乎杂乱无章的数据中找出其近似的规律性,即设法构造一条曲线反映所给数据点的总趋势,以消除其局部的波动。这种数据处理的方法在实际问题中用得很广。下面介绍几种常用的曲线拟合方法。

3.6.1 直线拟合法

直线拟合法是最简单的一种曲线拟合方法,即构造一条直线 $y=a+bx$,使采样点数据 $(x_i, y_i)(i=1,2,\cdots,n)$ 之间的函数关系由下式近似比拟:

$$g(x_i)=a+bx_i,\quad (i=1,2,\cdots,n),\quad (n\gg 2)$$

采样数据点的数目 n 远大于待定系数(a 与 b)的数目,因此,拟合直线的构造本质上是一个解超定方程组的代数问题。

设 $\tilde{y}_i=g(x_i)=a+bx_i(i=1,2,\cdots,n)$ 表示按拟合直线 $y=a+bx$ 求得的近似值,$\tilde{y}_i$ 一般不同于实测值 y_i,两者之差 $y_i-\tilde{y}_i=e_i$ 称为残差。显然,e_i 越小,拟合质量越高。

关于拟合质量的优劣有三种衡量准则:

准则 1 “使残差的最大绝对值为最小”准则,即:

$$\max_{1\leqslant i\leqslant n}|e_i|=\min$$

准则 2 “使残差的绝对值之和为最小”准则,即:

$$\sum_{i=1}^{n}|e_i|=\min$$

准则 3 “使残差的平方和为最小”准则,即:

$$\sum_{i=1}^{n}e_i^2=\min$$

在这三种准则中,常用准则 3,用此准则所导出的曲线拟合方法就称为最小二乘法。

下面就讨论如何用准则 3 确定拟合直线 $y=a+bx$。

设给定采样数据点 $(x_i, y_i)(i=1,2,\cdots,n)$,构造一条直线 $g(x_i)=a+bx_i$,使总误差为:

$$Q=\sum_{i=1}^{n}e_i^2=\sum_{i=1}^{n}[y_i-g(x_i)]^2=\sum_{i=1}^{n}[y_i-(a+bx_i)]^2=\min$$

可见,残差 Q 是系数 a,b 的函数。因此,当

$$\frac{\partial Q}{\partial a}=-2\sum_{i=1}^{n}(y_i-a-bx_i)=0$$

和

$$\frac{\partial Q}{\partial b}=-2\sum_{i=1}^{n}x_i(y_i-a-bx_i)=0$$

时，残差 Q 最小。于是可推导出：

$$na+b\sum_{i=1}^{n}x_i=\sum_{i=1}^{n}y_i$$

以及

$$a\sum_{i=1}^{n}x_i+b\sum_{i=1}^{n}x_i^2=\sum_{i=1}^{n}x_iy_i$$

所以

$$a=\frac{\left(\sum x_i^2\right)\left(\sum y_i\right)-\left(\sum x_iy_i\right)\left(\sum x_i\right)}{n\left(\sum x_i^2\right)-\left(\sum x_i\right)^2} \tag{3.31}$$

$$b=\frac{n\left(\sum x_iy_i\right)-\left(\sum x_i\right)\left(\sum y_i\right)}{n\left(\sum x_i^2\right)-\left(\sum x_i\right)^2} \tag{3.32}$$

由(3.31)和(3.32)式即可确定拟合直线为：

$$y=a+bx$$

例3.10 求解下面的超定方程组：

$$\begin{cases}a+b=3\\2a-b=0.2\\a+3b=7\\3a+b=5\end{cases}$$

解 显然，该方程组中的 a、b 取任何值都不可能同时满足上面4个方程，只能用使每个方程的残差平方和最小的近似计算，才能够求解。为此，设每个方程出现的残差为：

$$\varepsilon_1=3-(a+b) \qquad \varepsilon_2=0.2-(2a-b)$$

$$\varepsilon_3=7-(a+3b) \qquad \varepsilon_4=5-(3a+b)$$

从该残差方程组中解出使残差平方和为最小的参数 (a,b)，作原方程的最小二乘解，即：

$$\begin{cases}15a+5b=25.4\\5a+12b=28.8\end{cases}$$

解之，得：

$$a=1.037,\quad b=1.968$$

采用直线拟合方法的前提条件是，数据之间的函数关系大致为直线关系。如果数据之间的函数关系根本不符合直线规律，则应该采用多项式拟合方法和其他拟合方法，如指数拟合等。

3.6.2 多项式曲线拟合法

对于给定的一组数据 $(x_i,y_i)(i=1,2,\cdots,n)$，构造一个 m 次多项式 $(m\ll n)$，即：

$$g_m(x)=a_0+a_1x+a_2x^2+\cdots+a_mx^m$$

或

$$g_m(x)=\sum_{j=0}^{m}a_jx^j \tag{3.33}$$

使得该多项式和给定的数据 $(x_i,y_i)(i=1,2,\cdots,n)$ 之间可近似成立如下函数关系：

$$\tilde{y}_i = g(x_i) = \sum_{j=0}^{n} a_j x_i^j = a_0 + a_1 x_i + \cdots + a_k x_i^k + \cdots + a_m x_i^m \quad (i=1,2,\cdots,n)$$

设 $\tilde{y}_i$ 表示按(3.33)式求得的近似值，一般不同于实测值 y_i，两者之差为 $e_i = y_i - \tilde{y}_i$，则有：

$$Q = \sum_{i=1}^{n} e_i^2 = \sum_{i=1}^{n} \left[y_i - \sum_{j=0}^{m} a_j x_i^j \right]^2$$

根据准则 3，有：

$$\frac{\partial Q}{\partial a_k} = 2\sum_{i=1}^{n}\left(y_i - \sum_{j=0}^{m} a_j x_i^j \right) \times x_i^k = 2\left[\sum_{i=1}^{n} x_i^k y_i - \sum_{i=1}^{n}\left(\sum_{j=0}^{m} a_j x_i^j \right) x_i^k \right] = 0$$

即

$$\sum_{i=1}^{n}(a_0 + a_1 x_i + \cdots + a_k x_i^k + \cdots + a_m x_i^m) x_i^k = \sum_{i=1}^{n} x_i^k y_i \quad (k=0,1,2,\cdots,m)$$

当 $k=0,1,2,\cdots,m$ 时，将上式展开，可得到下面一个关于 $a_i(i=0,1,\cdots,m)$ 的线性方程组：

$$\begin{cases} n \cdot a_0 + a_1 \sum_{i=1}^{n} x_i + \cdots + a_k \sum_{i=1}^{n} x_i^k + \cdots + a_m \sum_{i=1}^{n} x_i^m = \sum_{i=1}^{n} y_i \\ a_0 \sum_{i=1}^{n} x_i + a_1 \sum_{i=1}^{n} x_i^2 + \cdots + a_k \sum_{i=1}^{n} x_i^{k+1} + \cdots + a_m \sum_{i=1}^{n} x_i^{m+1} = \sum_{i=1}^{n} x_i y_i \\ \cdots \\ a_0 \sum_{i=1}^{n} x_i^m + a_1 \sum_{i=1}^{n} x_i^{m+1} + \cdots + a_k \sum_{i=1}^{n} x_i^{m+k} + \cdots + a_m \sum_{i=1}^{n} x_i^{2m} = \sum_{i=1}^{n} x_i^m y_i \end{cases} \tag{3.34}$$

方程组(3.34)通常称为正规方程组。其中，m 为多项式的次数，n 为采样点数。

可用Guass 消去法解此方程组，求得系数 $a_i(i=0,1,\cdots,m)$，得到拟合多项式(3.33)。但在实际问题中，当多项式的次数较高时，正规方程组的系数行列式会出现"病态"。因此，常采用正交多项式作为拟合曲线。

二次多项式曲线拟合，即抛物线拟合

$$y = a + bx + cx^2$$

应用比较常见，其系数的求解过程如下：

抛物线拟合的正规方程组为：

$$\begin{aligned} na + b\sum x_i + c\sum x_i^2 &= \sum y_i \\ a\sum x_i + b\sum x_i^2 + c\sum x_i^3 &= \sum x_i y_i \\ a\sum x_i^2 + b\sum x_i^2 + c\sum x_i^4 &= \sum x_i^2 y_i \end{aligned} \tag{3.35}$$

式中，$\sum$ 表示从 $i=1\sim n$ 累加。解此方程组即可求得系数 a,b,c。用MATLAB 矩阵运算方法，可以将(3.35)式表示成矩阵形式 $AC=B$，其中：

$$A = \begin{bmatrix} \sum x_i^0 & \sum x_i^1 & \sum x_i^2 \\ \sum x_i^1 & \sum x_i^2 & \sum x_i^3 \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 \end{bmatrix}, \quad C = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, \quad B = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \end{bmatrix}$$

在MATLAB 中用命令语句

$$C = A\backslash B$$

求解。求拟合多项式系数的一种更简单的方法是用命令 polyfit，即：

$$C = \text{polyfit}(x, y, n)$$

其中，x 和 y 是采样点数据，n 是拟合多项式最高阶次。

例 3.11　用二次多项式拟合下列数据，并画出数据点和拟合曲线图像。

x=[0.1,0.4,0.5,0.6,0.7,0.9]

y=[0.61,0.92,0.99,1.52,1.47,2.03]

解　用polyfit 函数求二次多项式的系数，并可绘出拟合曲线如图3.3 所示。其MATLAB程序如下：

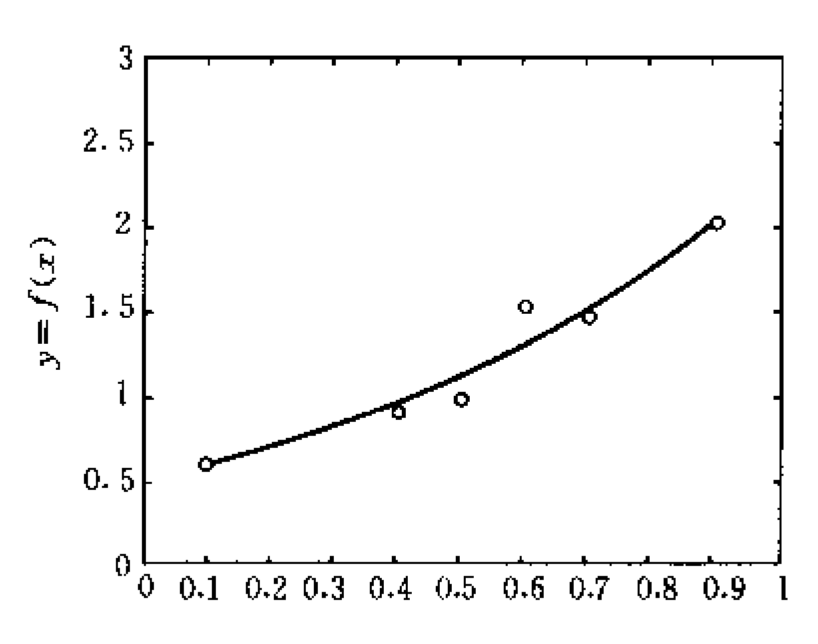


图 3.3　曲线拟合结果

```
clear,clf
x=[0.1,0.4,0.5,0.6,0.7,0.9];
y=[0.61,0.92,0.99,1.52,1.47,2.03];
plot(x,y,'o')
hold on
c=polyfit(x,y,2)
xx=x(1):0.1:x(length(x));
yy=polyval(c,xx);
plot(xx,yy)
axis([0,1,0,3])
xlabel('x ')
ylabel('y=f(x)')
c=
   1.26233766233766 0.549870129870132 0.525792207792207
```

3.6.3　指数曲线拟合法

对某一类型的数据可以用指数曲线

$$y=ae^{bx} \tag{3.36}$$

来进行拟合。其中 a,b 为待定系数($a>0$)。这类曲线拟合可以按下述步骤确定系数：

(1) 将 $y=ae^{bx}$ 两边取对数，得：

$$\ln y=\ln a+bx$$

(2) 令 $\ln y=Y,\ln a=A,b=B$，则可以得到一个直线方程：

$$Y=A+BX$$

(3) 用直线拟合的方法，其拟合的数据点为($\ln y,x$)，求出上述变换后的直线方程中的待

定系数 A 和 B。

(4) 根据第(2)步中所得的关系式,求出指数方程中的待定系数 a 和 b,即可得到指数曲线拟合方程 $y=ae^{bx}$。

一般来说,通过适当变换可以把某些非多项式曲线拟合转化为直线拟合或多项式拟合来求解。用上述通过将方程两边取对数的方法转化为直线进行拟合的曲线类型还有:

$$y=ae^{-bx}$$

$$y=ax^{b}$$

$$y=x^{t}(ax+b)$$

$$\vdots$$

检验拟合精度的几个指标为:

(1) 残差平方和 $Q=\sum(\tilde{y}_i-y_i)^2$,$Q$ 越小越好。

(2) 标准差 $\sigma_a=Q/(n-2)$,σ_a 越小越好。

(3) 相关系数 $R=1-\dfrac{\sum(\tilde{y}_i-y_i)^2}{\sum(y_i-y)^2}$,$R$ 越接近 1 越好。

在上述公式中,y_i 为样点 y 坐标的实测值,$\tilde{y}_i$ 为按拟合曲线计算所得的值,y 为 $\sum y_i/n$,n 为采样点数,式中的加法运算符 $\sum$ 都表示 $\sum\limits_{i=1}^{n}$。

检验时只需取上述指标中 1～2 个即可,一般用残差平方和 Q 和相关系数 R 两个指标来进行检验。

本 章 小 结

本章主要内容为插值方法和曲线拟合方法。

拉格朗日插值方法是一种常用的插值方法,但在插值节点数目过多时会产生龙格现象,尤其是在插值区间的两端。为避免发生龙格现象,常采用逐次插值方法、分段插值方法和埃尔米特插值方法等。逐次插值方法有埃特金插值法和牛顿插值法。埃特金插值法可以归结为线性插值的多次重复,不断提高插值多项式的阶次,提高插值精度。和埃特金插值法相比,牛顿插值法具有简洁的、承袭性的数学公式,在理论分析上十分方便。分段插值方法只要插值区间足够小,就可以达到任意指定的计算精度。埃尔米特插值法要求插值函数和被插值的函数在插值节点相切,其插值条件要比拉格朗日插值法高得多。

曲线拟合方法是研究如何利用离散的数据,拟合成为一条可以用某个函数或某几个函数来表达的问题。这种问题在计算机辅助设计(CAD)中用得很多。本章主要讨论直线拟合、多项式拟合和指数拟合等三种拟合方法。常采用残差的平方和是否最小作为曲线拟合准确度的评价标准。

本章主要掌握的内容是:

(1) 插值方法和曲线拟合方法的应用场合。

(2) 拉格朗日插值多项式的构成方法、插值基函数和插值余项。

(3) 插值过程中的龙格现象和防止措施。

(4) 各种插值公式的特点和应用问题。

(5) 判断离散数据插值计算精度的方法。

(6) 线性最小二乘拟合原理和直线拟合的方法。

习　题　三

3.1　已知 $y=f(x)$ 的函数表如下：

x:	1	3
y:	1	2

求线性插值多项式，并计算 $x=1.5$ 的值。

3.2　已知 $y=f(x)$ 的函数表如下：

x:	1	3	2
y:	1	2	-1

求抛物线插值多项式，并计算 $x=1.5$ 的近似值。

3.3　已知函数表如下：

x:	1.1275	1.1503	1.1735	1.9720
$f(x)$:	0.1191	0.13954	0.15932	0.17903

应用 Lagrange 插值公式计算 $f(1.1300)$ 的近似值。

3.4　求过点(0,1)，(1,2)，(2,3)的三点插值公式，请分析结果并给出结论。

3.5　就题 3.2 的数据，写出牛顿插值多项式，并计算 $x=1.5$ 的近似值。

3.6　写出分段线性插值法的算法框图。

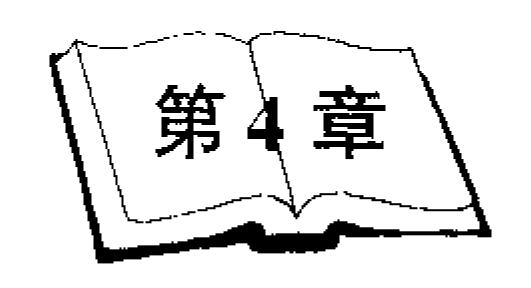
3.7　已知单调连续函数 $f(x)$ 在 $x=-1,2,3$ 时的值分别是 $-3,-1,4$，试用牛顿插值法计算 $f(x)$ 在区间 $[-1,3]$ 中的根。

3.8　用最小二乘法求解下列超定方程组：

$$\begin{cases}2x+4y=1\\3x-5y=3\\x+2y=6\\3x+y=7\end{cases}$$

3.9　用最小二乘法求形如 $y=a+bx^2$ 的多项式，使之与下列数据相拟合：

x:	19	25	31	38	44
y:	19.0	32.3	49.0	73.3	97.8



数值积分

4.1 引　　言

如果被积函数$f(x)$在区间$[a,b]$上连续且其原函数为$F(x)$，则可用Newton-Leibnitz(牛顿-莱布尼兹)公式

$$I=\int_a^b f(x)\mathrm{d}x=F(x)\Big|_a^b=F(b)-F(a)$$

来计算定积分I。但是在许多情形下，不能用Newton-Leibnitz公式来计算定积分。例如：

(1) $f(x)$不是用具体的数学表达式而是用一个数据表来表达的函数，即表格函数：

$$\begin{array}{lcccc} x: & x_1 & x_2 & \cdots & x_n \\ f(x): & f(x_1) & f(x_2) & \cdots & f(x_n) \end{array}$$

或者$f(x)$定义为某个微分方程的解，但该微分方程的解不能用解析法解出，而用数值方法给出的解同样是表格函数。

(2) 被积函数$f(x)$的原函数$F(x)$不能够用初等函数的有限形式来表示，例如：

$$f(x)=\frac{\sin x}{x},\sin x^2,\cos x^2,\frac{1}{\ln x},e^{-x^2},\sqrt{1+x^3},\cdots$$

显然，在上述两种情形下，是无法用Newton-Leibnitz公式进行计算的。另外，被积函数$f(x)$的原函数尽管能够用初等函数的有限形式表示，例如：

$$\int_{\sqrt{3}}^{x}\frac{\mathrm{d}x}{1+x^4}=\left\{\frac{1}{4\sqrt{2}}\ln\frac{x^2+\sqrt{2}x+1}{x^2-\sqrt{2}x+1}+\frac{1}{2\sqrt{2}}\left[\arctan(\sqrt{2}x+1)+\arctan(\sqrt{2}x-1)\right]\right\}_{\sqrt{3}}^{x}$$

和 $$\int_a^b x^2\sqrt{2x^2+3}\mathrm{d}x=\left\{\frac{1}{4}x^2\sqrt{2x^2+3}+\frac{3}{16}x\sqrt{2x^2+3}-\frac{9}{16\sqrt{2}}\ln(\sqrt{2}x+\sqrt{2x^2+3})\right\}_a^b$$

等，但由于表达式过于复杂，也不便于实际上的应用。

因此，在Newton-Leibnitz公式应用受到限制时，可以采用数值方法构造计算公式，即所谓的数值积分方法来解决定积分的计算问题。

4.2 数值积分方法

4.2.1 数值积分的基本思想

由高等数学可知，积分中值定理的表示形式为：

$$I=\int_a^b f(x)\mathrm{d}x=(b-a)f(\varepsilon) \tag{4.1}$$

其中，ε为积分区间(a,b)内的某个点，虽然从理论上给出了求定积分I的另一种解析方法，但

是，由于在一般情况下无法准确得到ε之值，$f(\varepsilon)$之值也就无从计算，所以仍然难以准确计算定积分I。因此，在实际问题中往往用数值方法构造某种近似公式来代替$f(\varepsilon)$，从而构造出相应的数值积分公式，即求积公式。为构造一般的求积公式，下面先来看一看在高等数学中见到过的下面几种简单的求积公式：

1）矩形求积公式

左矩形公式 $$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(a)=L \tag{4.2}$$

右矩形公式 $$\int_a^b f(x)\mathrm{d}x \approx (b-a)f(b)=L \tag{4.3}$$

中矩形公式 $$\int_a^b f(x)\mathrm{d}x \approx (b-a)f\left(\frac{a+b}{2}\right)=L \tag{4.4}$$

矩形求积公式是用区间$[a,b]$内的某一点的函数值来近似代替$f(\varepsilon)$而构成数值积分公式的。

2）梯形求积公式

$$\int_a^b f(x)\mathrm{d}x \approx \frac{1}{2}(b-a)[f(a)+f(b)]=T \tag{4.5}$$

梯形求积公式是用区间$[a,b]$内的两个端点的函数值的平均值来近似代替$f(\varepsilon)$而构成数值积分公式的。

3. Simpson（辛普生）求积公式

$$\int_a^b f(x)\mathrm{d}x \approx \frac{1}{6}(b-a)\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]=S \tag{4.6}$$

Simpson 求积公式是用区间$[a,b]$内三个点的函数值的加权平均值来近似代替$f(\varepsilon)$而构成数值积分公式的。

图4.1给出了这三种求积公式的示意图。图中阴影部分面积表示用数值方法计算的定积分近似值，并称其曲边为数值曲线。它是函数曲线的近似。在这三种简单的求积公式中，Simpson 求积公式的精度是最高的，但仍然和准确值之间存在误差。怎样提高求积公式的计算精度呢？可以从上述三种简单求积公式的构造中得到启发：在区间$[a,b]$内，能否用更多点的函数值的加权平均值构造一个精度更高的数值积分公式呢？答案是肯定的。因为在一般情形下，在区间$[a,b]$内，如果数值曲线上的点和函数曲线上的点重合越多、两条曲线越接近，数值积分的精度就应该越高。由此得到了数值积分的基本思想：在区间$[a,b]$内，用充分多的点的函数值的加权平均值来代替$f(\varepsilon)$，从而可构造出一般的求积公式。

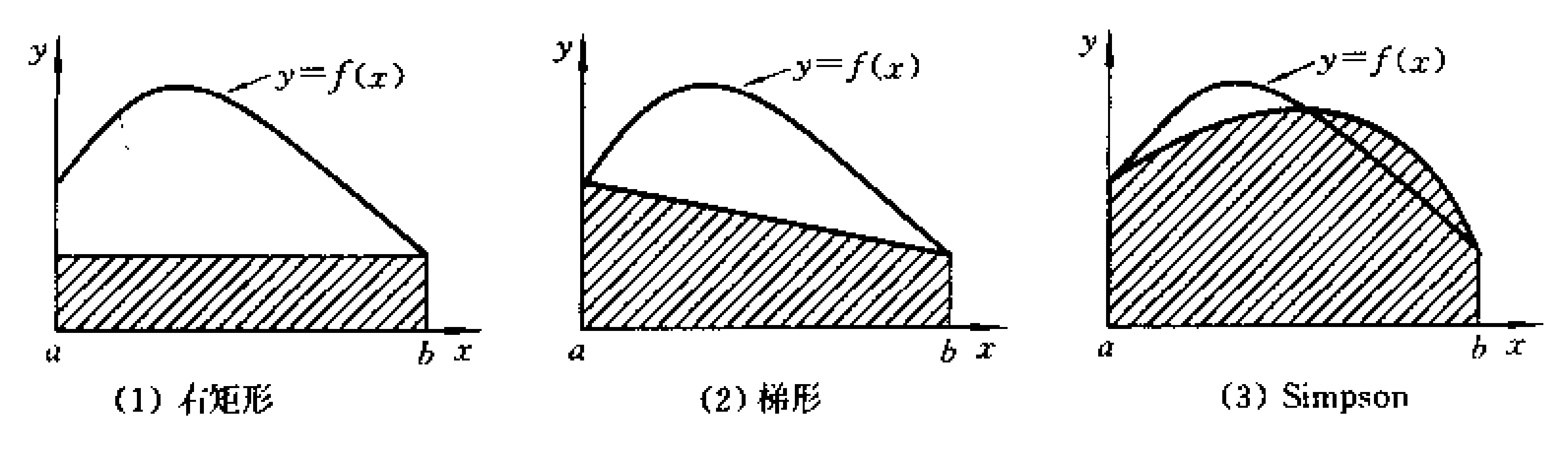


图4.1 右矩形、梯形和Simpson求积公式示意图

4.2.2 一般求积公式

若在区间$[a,b]$内，取$n+1$个节点$x_i(i=0,1,2,\cdots,n)$，用$f(x_i)(i=0,1,2,\cdots,n)$的加权平均值作为$f(\varepsilon)$的近似值，即：

$$f(\varepsilon) \approx \sum_{i=0}^{n} C_i f(x_i)$$

其中
$$\sum_{i=0}^{n} C_i = 1$$

将上式代入(4.1)式,有:

$$\int_a^b f(x)\mathrm{d}x = (b-a)\ f(\varepsilon) \approx (b-a) \sum_{i=0}^{n} C_i f(x_i)$$

或写成
$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{n} A_i f(x_i) \tag{4.7}$$

其中,$x_i(i=0,1,2,\cdots,n)$称为求积节点,$A_i(i=0,1,2,\cdots,n)$称为求积系数,A_i 只依赖于求积节点和积分区间,而与求积函数 $f(x)$ 无关。式(4.7)称为一般求积公式或机械求积公式。

这种用若干求积节点的函数值的加权平均值来计算定积分的方法,称为机械求积方法。机械求积方法将积分中求原函数 $F(x)$ 的问题转化为求节点 x_i 处的函数值 $f(x_i)$ 的问题,使积分问题的计算得到大大简化。

机械求积公式是一般形式的数值积分公式,在实际应用中,必须首先解决两个基本问题:求积公式的精度与求积节点数有什么关系?如何确定求积公式中的节点和求积系数?下面就具体研究这两个问题。

4.2.3　求积公式的代数精度

求积公式的精度可以用余项 R 来表示,即:

$$\int_a^b f(x)\mathrm{d}x = \sum_{k=0}^{n} A_i f(x_i) + R$$

一般地,R 越小,表明求积公式的精度越高。但是,当被积函数 $f(x)$ 是多项式时,R 却表现出某种特性,即某一确定的求积公式,对某些多项式的积分是精确的,而对另外一些多项式的积分是不精确的。下面就通过具体例子来说明这一现象。

例 4.1　用梯形公式和 Simpson 公式分别求 $f(x)=1,x,x^2,x^3,x^4$ 时下列定积分之值,并给出结论。

$$I = \int_0^1 f(x)\mathrm{d}x$$

解　计算定积分的梯形公式和 Simpson 公式分别为:

$$T = \frac{1}{2}[f(0)+f(1)]$$

$$S = \frac{1}{6}\left[f(0)+4f\left(\frac{1}{2}\right)+f(1)\right]$$

将计算结果列成表 4.1(作为对比,准确值 I 用 Newton-Leibnitz 公式计算)。

表 4.1　计算例 4.1 题的结果

$f(x)$	1	x	x^2	x^3	x^4
I	1	1/2	1/3	1/4	1/5
T	1	1/2	1/2	1/2	1/2
S	1	1/2	1/3	1/4	5/24

从计算结果中可以得出如下结论：

梯形公式对最高次数不超过1的多项式的积分，计算结果是精确的，而对最高次数超过1的多项式的积分，计算结果是不精确的；Simpson 公式对最高次数不超过3的多项式的积分，计算结果是精确的，而对最高次数超过3的多项式的积分，计算结果是不精确的；用Simpson公式计算定积分要比梯形公式精确些。

为了更好地刻画求积公式精度的这种性质，就需要引入下面代数精度的概念：

定义4.1　如果求积公式(4.7)对于一切次数小于和等于n的多项式的积分是准确的，而对次数为$n+1$的多项式的积分至少有一个是不准确的，则称该求积公式具有n次代数精度，或称该公式是n阶的。

在具体考察一个求积公式的代数精度时，常常用以下方法：用某求积公式计算被积函数$f(x)=x^k$的积分，如果在$k=0,1,2,\cdots,n$时均是准确的，而在$k=n+1$时不是准确的，那么就可以确定该求积公式的代数精度是n次而不必再继续考察其对所有的$n+1$次多项式的积分是否准确成立(可以根据定积分的性质加以证明)。

在一般情形下，就用代数精度来衡量一个求积公式的精度。

例4.2　考察梯形公式的代数精度。

精确值计算用　$I=\int_a^b f(x)\mathrm{d}x=F(b)-F(a)$

近似值计算用　$T=\frac{1}{2}(b-a)[f(a)+f(b)]$

解　(1) 设$f(x)=1$，则

$$I=\int_a^b \mathrm{d}x=b-a$$

$$T=\frac{1}{2}(b-a)[f(a)+f(b)]=\frac{1}{2}(b-a)[1+1]=b-a$$

$T=I$，梯形公式对一切零次多项式的积分计算是精确的。

(2) 设$f(x)=x$，则

$$I=\int_a^b x\mathrm{d}x=\frac{1}{2}(b^2-a^2)$$

$$T=\frac{1}{2}(b-a)\ [f(a)+f(b)]=\frac{1}{2}(b-a)[b+a]=\frac{1}{2}(b^2-a^2)$$

$T=I$，梯形公式对一切一次多项式的积分计算是精确的。

(3) 设$f(x)=x^2$，则

$$I=\int_a^b x^2\mathrm{d}x=\frac{1}{3}(b^3-a^3)$$

$$T=\frac{1}{2}(b-a)[f(a)+f(b)]=\frac{1}{2}(b-a)[b^2+a^2]$$

$T\neq I$，梯形公式对二次多项式的积分计算不是精确的。

由代数精度的定义可知，梯形求积公式的代数精度为1次。

关于求积公式的精度与求积节点数的关系，有如下定理：

定理4.1　对于任意给定的$n+1$个互异的节点：

$$a=x_0<x_1<x_2<\cdots<x_n=b$$

总存在系数$A_0,A_1,A_2,\cdots,A_n$，使得求积公式(4.7)至少具有n次代数精度。

证明 以$n+1$个互异的节点$a=x_0<x_1<x_2<\cdots<x_n=b$为插值节点的Lagrange插值多项式为：

$$p_n(x)=\sum_{k=0}^{n}l_k(x)\cdot f(x_k)$$

其中
$$l_k(x)=\prod_{\substack{i=0\\i\neq k}}^{n}\frac{x-x_i}{x_k-x_i}$$

它们都是n次多项式。取求积系数

$$A_k=\int_a^b l_k(x)\mathrm{d}x$$

构成求积公式(4.7)，A_k只依赖于求积节点和积分区间，而与求积函数$f(x)$无关。由插值余项公式，有：

$$\begin{aligned}\int_a^b f(x)\mathrm{d}x&=\int_a^b p_n(x)\mathrm{d}x+\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\mathrm{d}x\\&=\int_a^b\sum_{k=0}^{n}l_k(x)\cdot f(x_k)\mathrm{d}x+\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)\prod_{i=0}^{n}(x-x_i)\mathrm{d}x\\&=\sum_{k=0}^{n}\int_a^b l_k(x)\mathrm{d}x\cdot f(x_k)+\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)\prod_{i=0}^{n}(x-x_i)\mathrm{d}x\\&=\sum_{k=0}^{n}A_k\cdot f(x_k)+\frac{1}{(n+1)!}\int_a^b f^{(n+1)}(\xi)\prod_{i=0}^{n}(x-x_i)\mathrm{d}x\end{aligned}$$

显然，对于一切$f(x)=x^k(k=0,1,2,\cdots,n)$，总有

$$\int_a^b f(x)\mathrm{d}x=\sum_{k=0}^{n}A_k\cdot f(x_k)$$

准确成立，故求积公式(4.7)至少具有n次代数精度。

4.2.4 求积公式的构造方法

1. 待定系数法

待定系数法就是以代数精度为标准来构造求积公式(4.7)。

例如，构造一个形如(4.7)式的求积公式，使其具有一次代数精度。此求积公式的具体形式应为：

$$\int_a^b f(x)\mathrm{d}x=A_0f(a)+A_1f(b)$$

利用代数精度的概念，可得以下两个方程式：

(1) 令$f(x)=1$，则有$A_0+A_1=b-a$。

(2) 令$f(x)=x$，则有$A_0a+A_1b=(b^2-a^2)/2$。

解之得
$$A_0=A_1=(b-a)/2$$

因此，以区间两端点a、b为节点的两点求积公式为：

$$T=\frac{b-a}{2}[f(a)+f(b)]$$

该式即为梯形求积公式。也就是说，梯形求积公式仅具有一次代数精度。

一般地，对于$n+1$求积节点$x_i(i=0,1,2,\cdots,n)$，若构造一个求积公式(4.7)，使其具有n

次代数精度，则必须使其当

$$f(x)=1,x,x^2,\cdots,x^n$$

时均能准确成立，即可得到如下的方程组：

$$\begin{cases}A_0+A_1+\cdots+A_n=b-a\\A_0x_0+A_1x_1+\cdots+A_nx_n=(b^2-a^2)/2\\\quad\cdots\\A_0x_0^n+A_1x_1^n+\cdots+A_nx_n^n=(b^{n+1}-a^{n+1})/(n+1)\end{cases}\tag{4.8}$$

对于方程组(4.8)：

(1) 若 $x_i(i=0,1,2,\cdots,n)$ 已给定，则方程组(4.8)只有 $n+1$ 个待定系数 $A_0,A_1,\cdots,A_n$，是一个线性方程组；当 $x_i(i=0,1,2,\cdots,n)$ 互异时，其系数矩阵行列——范德蒙行列式不为零，故方程组有惟一解，这样就可求出全部的系数 $A_0,A_1,\cdots,A_n$，而且构造的求积公式(4.7)至少具有 n 次代数精度。

当 $x_i(i=0,1,2,\cdots,n)$ 为等距节点时，构造的求积公式称为Newton-Cotes(牛顿-柯特斯)公式。

(2) 若 A_i 和 $x_i(i=0,1,2,\cdots,n)$ 均为待定，则方程组(4.8)具有 $(2n+2)$ 个待定系数 $A_0,A_1,\cdots A_n$ 和 $x_0,x_1,\cdots,x_n$。为求出这些系数，必须使求积公式(4.7)当 $f(x)=x^k(k=0,1,2,\cdots,2n+1)$ 时均准确成立，即使之具有 $2n+1$ 次代数精度，从而得到 $2n+2$ 个方程构造的高阶非线性方程组：

$$\begin{cases}A_0+A_1+\cdots+A_n=b-a\\A_0x_0+A_1x_1+\cdots+A_nx_n=(b^2-a^2)/2\\\quad\cdots\\A_0x_0^{2n+1}+A_1x_1^{2n+1}+\cdots+A_nx_n^{2n+1}=(b^{2n+2}-a^{2n+2})/(2n+2)\end{cases}\tag{4.9}$$

解方程组(4.9)得到 A_i 和 $x_i(i=0,1,2,\cdots,n)$，这样构造的求积公式称为Guass(高斯)求积公式。

例4.3　设有下列近似求积公式：

$$\int_{-1}^{1}f(x)\mathrm{d}x\approx Af(-1)+Df(-1/2)+Bf(0)+Ef(1/2)+Cf(1)$$

试用待定系数法求出各系数，并求该求积公式的最高代数精度。

解　求积公式有5个求积节点，故分别取 $f(x)=1,x,x^2,x^3,x^4$ 进行计算。

由 $f(x)=1$，$\int_{-1}^{1}f(x)\mathrm{d}x=\int_{-1}^{1}\mathrm{d}x=2$，得 $A+D+B+E+C=2$；

由 $f(x)=x$，得 $\int_{-1}^{1}f(x)\mathrm{d}x=\int_{-1}^{1}x\mathrm{d}x=0$，得 $-A-\frac{1}{2}D+0B+\frac{1}{2}E+C=0$；

由 $f(x)=x^2$，$\int_{-1}^{1}f(x)\mathrm{d}x=\int_{-1}^{1}x^2\mathrm{d}x=\frac{2}{3}$，得 $A+\frac{1}{2^2}D+0B+\frac{1}{2^2}E+C=\frac{2}{3}$；

由 $f(x)=x^3$，$\int_{-1}^{1}f(x)\mathrm{d}x=\int_{-1}^{1}x^3\mathrm{d}x=0$，得 $-A-\frac{1}{2^3}D+0B+\frac{1}{2^3}E+C=0$；

由 $f(x)=x^4$，$\int_{-1}^{1}f(x)\mathrm{d}x=\int_{-1}^{1}x^4\mathrm{d}x=\frac{2}{5}$，得 $A+\frac{1}{2^4}D+0B+\frac{1}{2^4}E+C=\frac{2}{5}$。

最后得到一个线性方程组：

$$\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -1 & -\frac{1}{2} & 0 & \frac{1}{2} & 1 \\ 1 & \frac{1}{4} & 0 & \frac{1}{4} & 1 \\ -1 & -\frac{1}{8} & 0 & \frac{1}{8} & 1 \\ 1 & \frac{1}{16} & 0 & \frac{1}{16} & 1 \end{bmatrix} \begin{bmatrix} A \\ D \\ B \\ E \\ C \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{2}{3} \\ 0 \\ \frac{2}{5} \end{bmatrix}, \quad \text{解之,得} \quad \begin{bmatrix} A \\ D \\ B \\ E \\ C \end{bmatrix} = \begin{bmatrix} \frac{7}{45} \\ \frac{32}{45} \\ \frac{4}{15} \\ \frac{32}{45} \\ \frac{7}{45} \end{bmatrix}$$

所以

$$\int_{-1}^{1} f(x)\mathrm{d}x \cong \frac{7}{45}f(-1)+\frac{32}{45}f\left(-\frac{1}{2}\right)+\frac{4}{15}f(0)+\frac{32}{45}f\left(\frac{1}{2}\right)+\frac{7}{45}f(1)$$

可以验证,该求积公式对 $f(x)=x^5$ 是准确成立的,而对 $f(x)=x^6$ 不能准确成立,故该求积公式的最高精度为5次。

思考题　如果用 $f(x)=x^2,x^3,x^4,x^5,x^6$ 来确定上述积分公式的系数,是否可行?是否能得到代数精度至少为6次的求积公式?

2. 插值公式构造法

在 $I=\int_a^b f(x)\mathrm{d}x$ 中,如果用Lagrange插值多项式

$$p_n(x)=\sum_{k=0}^{n} l_k(x)\cdot f(x_k)$$

其中

$$l_k(x)=\prod_{\substack{i=0\\ i\neq k}}^{n}\frac{x-x_i}{x_k-x_i}$$

来近似地代替 $f(x)$,则有:

$$\int_a^b f(x)\mathrm{d}x \cong \int_a^b p_n(x)\mathrm{d}x=\int_a^b \sum_{k=0}^{n} l_k(x)f(x_k)\mathrm{d}x=\sum_{k=0}^{n}\int_a^b l_k(x)\mathrm{d}x\cdot f(x_k)$$

令

$$A_k=\int_a^b l_k(x)\mathrm{d}x \tag{4.10}$$

得

$$\int_a^b f(x)\mathrm{d}x \cong \int_a^b p_n(x)\mathrm{d}x=\sum_{k=0}^{n} A_k\cdot f(x_k) \tag{4.11}$$

在求积公式(4.7)中,如果求积系数 $A_k(k=0,1,2,\cdots,n)$ 由(4.10)式计算得到,则称该求积公式为插值型的。相应地公式(4.11)称为插值型求积公式。

定理4.2　由 $n+1$ 个求积节点构造的求积公式(4.7)至少有 n 次代数精度的充要条件是该公式是插值型的。

证明　定理4.1已经证明了充分性,现只需证明必要性。

插值基函数 $l_k(x)(k=0,1,2,\cdots,n)$ 是 x 的 n 次多项式,且

$$l_k(x_i)=\begin{cases}0, & i\neq k\\ 1, & i=k\end{cases}$$

故当 $f(x)=l_k(x)$ 时,求积公式(4.7)式能够精确成立,即:

$$\int_a^b f(x)\mathrm{d}x=\int_a^b l_k(x)\mathrm{d}x=\sum_{i=0}^{n} A_i\cdot l_k(x_i)\quad (k=0,1,2,\cdots,n)$$

而
$$\sum_{i=0}^{n} A_i \cdot l_k(x_i) = A_0 \times 0 + \cdots A_k \times 1 + \cdots A_n \times 0 = A_k$$

故
$$\int_a^b l_k(x)\mathrm{d}x = A_k \quad (k=0,1,2,\cdots,n)$$

所以,求积公式(4.7)是插值型的。

例4.4 考察下面求积公式的代数精度。

$$I=\int_{-1}^{1} f(x)\mathrm{d}x \approx \frac{1}{2}[f(-1)+2f(0)+f(1)]=L$$

解 设$f(x)=1$,则:

$$I=\int_{-1}^{1}\mathrm{d}x = 1-(-1)=2 \qquad L=\frac{1}{2}(b-a)[1+2+1]=2=I$$

设$f(x)=x$,则:

$$I=\int_{-1}^{1} x\mathrm{d}x=\frac{1}{2}(1^2-1^2)=0 \qquad L=\frac{1}{2}(b-a)[-1+2\times 0+1]=0=I$$

设$f(x)=x^2$,则:

$$I=\int_{-1}^{1} x^2\mathrm{d}x = \frac{1}{3}(1^3-(-1))^3=\frac{2}{3} \qquad L=\frac{1}{2}(b-a)[1+2\times 0+1]=1\neq I$$

由代数精度的定义可知,该求积公式的代数精度为1次。

应该注意,具有3个求积节点的求积公式的代数精度不一定高于2次,其原因是该求积公式不是插值型的。对于插值型求积公式,有:

$$\sum_{k=0}^{n} A_k = b-a$$

应用求积系数的这种关系式,可以检查计算求积系数的正确性。

从定理4.1和4.2可知,由于$n+1$个积分节点构造的求积公式(4.7)不一定具有大于等于n次的代数精度,故只有该公式是插值型的,才具有大于等于n次的代数精度。

最后,求积公式(4.7)的求积系数$A_0,A_1,A_2,\cdots,A_n$,可以由方程组(4.8)确定,也可以由式(4.10)计算,两者是完全一致的。

4.3 Newton-Cotes(牛顿 - 柯特斯)求积公式

4.3.1 Newton-Cotes 公式的一般形式

对于$I=\int_a^b f(x)\mathrm{d}x$,将区间$[a,b]$分成$n$等份,步长为:

$$h=(b-a)/n$$

故求积节点$x_k=a+kh,(k=0,1,2,\cdots,n)$,通过这$n+1$个节点构造一个$n$次代数多项式$p_n(x)\cong f(x)$。令

$$x=a+th \quad (t=0,1,2,\cdots,k,\cdots,n)$$

作积分变换
$$\mathrm{d}x=h\mathrm{d}t$$

$$x=a,t=0; \quad x=b,t=n$$

由(4.10)式计算求积系数$A_k(k=0,1,2,\cdots,n)$,即:

$$A_k=\int_a^b l_k(x)\mathrm{d}x=\int_a^b\prod_{\substack{j=0\\j\neq k}}^{n}\frac{x-x_j}{x_k-x_j}\mathrm{d}x=\int_0^n\prod_{\substack{j=0\\j\neq k}}^{n}\frac{(a+th)-(a+jh)}{(a+kh)-(a+jh)}h\mathrm{d}t$$

$$=\int_0^n\prod_{\substack{j=0\\j\neq k}}^{n}\frac{t-j}{k-j}h\mathrm{d}t=h\int_0^n\prod_{\substack{j=0\\j\neq k}}^{n}\frac{t-j}{k-j}\mathrm{d}t$$

因为

$$\prod_{\substack{j=0\\j\neq k}}^{n}(k-j)=k(k-1)\cdots(k-(k-1))\cdot(k-(k+1))\cdots(k-n)$$

$$=k(k-1)\cdots1\cdot(-1)(-2)\cdots(-(n-k))=k!\ (-1)^{n-k}(n-k)!$$

所以

$$A_k=\frac{h(-1)^{n-k}}{k!\ (n-k)!}\int_0^n\prod_{\substack{j=0\\j\neq k}}^{n}(t-j)\mathrm{d}t$$

令

$$C_k=\frac{A_k}{b-a}=\frac{A_k}{nh}=\frac{(-1)^{n-k}}{nk!\ (n-k)!}\int_0^n\prod_{\substack{j=0\\j\neq k}}^{n}(t-j)\mathrm{d}t \tag{4.12}$$

称C_k为Cotes(柯特斯)系数。故

$$\int_a^b f(x)\mathrm{d}x\cong(b-a)\sum_{k=0}^{n}C_kf(x_k) \tag{4.13}$$

(4.13)式即为n阶的Newton-Cotes公式的一般形式。

如果给定n,则C_k可由(4.12)式计算出来。表4.2列出了$n=1\sim10$的Cotes系数。应该注意:

(1) 表中的系数c_k要乘倍率K,才是公式(4.12)和(4.13)中的C_k。例如,当$n=3$时,$C_0=1/8$,$C_1=3/8$,$C_2=3/8$,$C_3=1/8$。

(2) 当n大于等于6时,表中未列出的系数可根据对称性得到。例如,当$n=6$时,$C_6=41/840$。

根据上述规则,可以写出$n=4$的Newton-Cotes公式如下:

$$\int_a^b f(x)\mathrm{d}x\cong\frac{1}{90}(b-a)[7f(x_0)+32f(x_1)+12f(x_2)+32f(x_3)+7f(x_4)]$$

其中,$x_0=a$,$x_4=b$。

表4.2　Cotes系数

n	K	c_0	c_1	c_2	c_3	c_4	c_5	…
1	1/2	1	1					
2	1/6	1	4	1				
3	1/8	1	3	3	1			
4	1/90	7	32	12	32	7		
5	1/228	19	75	50	50	75	19	
6	1/840	41	216	27	272	27	216	…
7	1/17280	751	3577	1323	2989	2989	1323	…
8	1/28350	989	5888	−928	10496	−4540	10496	…
9	1/89600	2857	15741	1080	19344	5778	5788	…
10	1/5996752	16067	106300	−48525	272400	−260550	427368	…

4.3.2 Newton-Cotes 公式的稳定性

在Newton-Cotes 公式(4.13)中，设计算函数值$f(x_k)(k=1,2,\cdots,n)$所产生的舍入误差为$\varepsilon_k(k=1,2,\cdots,n)$，则用Newton-Cotes 公式计算积分的误差为：

$$\eta=(b-a)\sum_{k=0}^{n}C_kf(x_k)-(b-a)\sum_{k=0}^{n}C_k[f(x_k)+\varepsilon_k]=(b-a)\sum_{k=0}^{n}C_k\varepsilon_k$$

令$\varepsilon=\max\limits_{0\leqslant k\leqslant n}|\varepsilon_k|$，当Cotes 系数全为正时($n\leqslant 7,n=9$)，则

$$|\eta|\leqslant(b-a)\sum_{k=0}^{n}|C_k\varepsilon_k|\leqslant(b-a)\varepsilon\sum_{k=0}^{n}|C_k|=(b-a)\varepsilon$$

所以，Newton-Cotes 公式是数值稳定的。如果Cotes 系数有正有负，则

$$\sum_{k=0}^{n}|C_k|>1$$

随着n的增大，该值变得越来越大，并且对$|\eta|$的影响越来越大，容易出现数值不稳定现象。因此，高阶($n\geqslant 8$)的Newton-Cotes 公式在实际中很少应用。本章只讨论几种低阶的Newton-Cotes 公式的具体形式。

4.3.3 截断误差与代数精度

1. 截断误差

Newton-Cotes 公式是插值型求积公式，故其余项R是Lagrange 插值多项式余项的积分：

$$R=\int_a^b f(x)\mathrm{d}x-\sum_{i=0}^{n}A_if(x_i)=\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\mathrm{d}x,\quad \xi\in[a,b]$$

即

$$R=\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\mathrm{d}x,\quad \xi\in[a,b]\tag{4.14}$$

作积分变换$x=a+th,\mathrm{d}x=h\mathrm{d}t,x=a,t=0;x=b,t=n$，有：

$$R=\int_0^n\frac{f^{(n+1)}(\xi)}{(n+1)!}(a+th-a-0h)(a+th-a-1h)\cdots(a+th-a-nh)h\mathrm{d}t$$

$$=\int_0^n\frac{f^{(n+1)}(\xi)}{(n+1)!}(t-0)(t-1)\cdots(t-n)h^{n+2}\mathrm{d}t,\quad \xi\in[a,b]$$

故

$$R=\frac{h^{n+2}}{(n+1)!}\int_0^n f^{(n+1)}(\xi)\prod_{k=0}^{n}(t-k)\mathrm{d}t,\quad \xi\in[a,b]\tag{4.15}$$

在推导几个低阶Newton-Cotes 公式的截断误差时，会用到广义积分中值定理。即：

定理4.3　设函数$g(x)$及$f(x)g(x)$在$[a,b]$上可积，函数$f(x)$在$[a,b]$上连续，且$m\leqslant f(x)\leqslant M$($m,M$为常数)，函数$g(x)$在$[a,b]$上不变号，即始终有$g(x)\leqslant 0$或$g(x)\geqslant 0$，则

$$\int_a^b f(x)g(x)\mathrm{d}x=f(\xi)\int_a^b g(x)\mathrm{d}x,\quad \xi\in[a,b]$$

2. 代数精度

定理4.4　具有$n+1$个求积节点的Newton-Cotes 公式在n为偶数时，其代数精度为$n+1$次。

证明　设$f(x)$为$n+1$次多项式，即：

$$f(x)=\sum_{i=0}^{n+1}a_i x^i$$

将其 $n+1$ 次导数 $f^{(n+1)}(x)=(n+1)!\ a_{n+1}$ 代入(4.15)式,得:

$$R=\frac{h^{n+2}}{(n+1)!}\int_0^n f^{(n+1)}(\xi)\prod_{k=0}^{n}(t-k)\mathrm{d}t=a_{n+1}h^{n+2}\int_0^n\prod_{k=0}^{n}(t-k)\mathrm{d}t \quad \xi\in[a,b]$$

令 $n=2m$(m 为整数),作积分变换 $u=t-m,\mathrm{d}u=\mathrm{d}t;t=0,u=-m;t=2m,u=m$。得

$$\int_0^n\prod_{k=0}^{n}(t-k)\mathrm{d}t=\int_{-m}^{m}\prod_{k=0}^{n}(u+m-k)\mathrm{d}u$$

因为
$$\begin{aligned}\prod_{k=0}^{n}(-u+m-k)&=(-u+m)(-u+m-1)\cdots(-u)\cdots(-u-m+1)(-u-m)\\&=(-1)^{2m+1}(u-m)(u-m+1)\cdots u\cdots(u+m-1)(u+m)\\&=(-1)^{2m+1}(u+m)(u+m-1)\cdots u\cdots(u-m+1)(u-m)\\&=-\prod_{k=0}^{n}(u+m-k)\end{aligned}$$

为奇函数,所以

$$R=a_{n+1}h^{n+2}\int_0^n\prod_{k=0}^{n}(t-k)\mathrm{d}t=a_{n+1}h^{n+2}\int_{-m}^{m}\prod_{k=0}^{n}(u+m-k)\mathrm{d}u=0$$

当 n 为偶数时,Newton-Cotes 公式对 $n+1$ 次多项式是准确成立的,故其代数精度为 $n+1$ 次。

4.3.4 低阶的 Newton-Cotes 公式

1. 梯形求积公式

当区间$[a,b]$分成 1 等分,即 $n=1$ 时,$h=(b-a)$,Cotes 系数计算如下:

$$C_0=\frac{(-1)^1}{1\cdot 0!\ \cdot(1)!}\int_0^1\prod_{\substack{j=0\\j\neq k}}^{1}(t-j)\mathrm{d}t=-\int_0^1(t-1)\mathrm{d}t=\frac{1}{2}$$

$$C_1=\frac{(-1)^0}{1\cdot 1!\ \cdot(0)!}\int_0^1\prod_{\substack{j=0\\j\neq k}}^{1}(t-j)\mathrm{d}t=\int_0^1 t\mathrm{d}t=\frac{1}{2}$$

得梯形求积公式:

$$\int_a^b f(x)\mathrm{d}x\cong\frac{1}{2}(b-a)[f(a)+f(b)]=\frac{1}{2}h[f(a)+f(b)]=T$$

当 $f(x)$在$[a,b]$上有连续的二阶导数时,梯形求积公式的截断误差为:

$$R_T=\int_a^b\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{i=0}^{n}(x-x_i)\mathrm{d}x=\int_a^b\frac{f^{(2)}(\xi)}{2}(x-a)(x-b)\mathrm{d}x,\quad \xi\in[a,b]$$

又在$[a,b]$上,始终有$(x-a)(x-b)\leqslant 0$,故根据定理 4.3 有:

$$R_T=\int_a^b\frac{f^{(2)}(\xi)}{2}(x-a)(x-b)\mathrm{d}x=\frac{f^{(2)}(\eta)}{2}\int_a^b(x-a)(x-b)\mathrm{d}x$$

故
$$R_T=-\frac{1}{12}(b-a)^3f^{(2)}(\eta)=-\frac{1}{12}h^3f^{(2)}(\eta)\quad \eta\in[a,b] \tag{4.16}$$

2. Simpson 求积公式

当区间$[a,b]$分成 2 等分,即 $n=2$ 时,$h=(b-a)/2$,Cotes 系数计算如下:

$$C_0=\frac{(-1)^2}{2\cdot 0!\ \cdot(2)!}\int_0^2\prod_{\substack{j=0\\j\neq k}}^{2}(t-j)\mathrm{d}t=\int_0^2(t-1)(t-2)\mathrm{d}t=\frac{1}{6}$$

$$C_1=\frac{(-1)^1}{2\cdot 1!\ \cdot(2)!}\int_0^2\prod_{\substack{j=0\\j\neq k}}^{2}(t-j)\mathrm{d}t=-\int_0^2 t(t-2)\mathrm{d}t=\frac{2}{3}$$

$$C_2=\frac{(-1)^0}{2\cdot 2!\ \cdot(0)!}\int_0^2\prod_{\substack{j=0\\j\neq k}}^{2}(t-j)\mathrm{d}t=\int_0^2 t(t-1)\mathrm{d}t=\frac{1}{6}$$

得Simpson 求积公式如下：

$$\int_a^b f(x)\mathrm{d}x\cong\frac{1}{6}(b-a)\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]=\frac{1}{3}h\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]=S$$

当$f(x)$在$[a,b]$上有连续的四阶导数时，Simpson 求积公式的截断误差为：

$$R_S=\int_a^b\frac{f^{(3)}(\xi)}{3!}(x-a)\left(x-\frac{a+b}{2}\right)(x-b)\mathrm{d}x\quad \xi\in[a,b]$$

$$\begin{aligned}R_S&=\int_a^b f[x,x_0,x_1,x_2](x-a)\left(x-\frac{a+b}{2}\right)(x-b)\mathrm{d}x\\&=\int_a^b f[x,x_0,x_1,x_2]\mathrm{d}\left(\frac{1}{4}(x-a)^2(x-b)^2\right)\\&=\frac{1}{4}f[x,x_0,x_1,x_2](x-a)^2(x-b)^2\Big|_a^b-\int_a^b\frac{1}{4}(x-a)^2(x-b)^2f'[x,x_0,x_1,x_2]\mathrm{d}x\\&=-\int_a^b\frac{1}{4}(x-a)^2(x-b)^2f[x,x,x_0,x_1,x_2]\mathrm{d}x\\&=-\frac{1}{4}f[\xi,\xi,x_0,x_1,x_2]\int_a^b(x-a)^2(x-b)^2\mathrm{d}x\end{aligned}$$

故

$$R_S=-\frac{1}{2880}(b-a)^5f^{(4)}(\eta)=-\frac{1}{90}h^5f^{(4)}(\eta)\quad \eta\in[a,b]\tag{4.17}$$

3. Cotes 求积公式

当区间$[a,b]$分成4 等分，即$n=4$ 时，$h=(b-a)/4$，同样可以计算出Cotes 系数及其截断误差为：

$$\int_a^b f(x)\mathrm{d}x\cong\frac{1}{90}(b-a)\left[7f(a)+32f(a+h)+12f\left(\frac{a+b}{2}\right)+32f(a+3h)+7f(b)\right]=C$$

其中，$h=\dfrac{b-a}{4}$。当$f(x)$在$[a,b]$上有连续的六阶导数时，有：

$$R_C=-\frac{8}{945}\left(\frac{b-a}{4}\right)^7f^{(6)}(\eta)=-\frac{8}{945}h^7f^{(6)}(\eta),\quad \eta\in[a,b]\tag{4.18}$$

一般称$n=4$ 时的Newton-Cotes 公式为Cotes 求积公式。

例4.5 用Newton-Cotes 公式计算下式：

$$I=\int_0^1\frac{\sin x}{x}\mathrm{d}x$$

解 用不同阶数的Newton-Cotes 公式计算，以便比较计算结果，即：

$$n=1, I\approx\frac{1}{2}[f(0)+f(1)]=\frac{1}{2}(1+0.8414709)=0.9207354$$

$$\begin{aligned}n=2, I&\approx\frac{1}{6}\left[f(0)+4f\left(\frac{1}{2}\right)+f(1)\right]\\&=\frac{1}{6}(1+4\times 0.9588511+0.8414709)=0.9461359\end{aligned}$$

$$n=3, I\approx\frac{1}{8}\left[f(0)+3f\left(\frac{1}{3}\right)+3f\left(\frac{2}{3}\right)+f(1)\right]=0.9461109$$

$$n=4, I\approx\frac{1}{90}\left[7f(0)+32f\left(\frac{1}{4}\right)+12f\left(\frac{1}{2}\right)+32f\left(\frac{3}{4}\right)+7f(1)\right]=0.9460830$$

$$n=5, I\approx\frac{19}{288}f(0)+\frac{25}{96}f\left(\frac{1}{5}\right)+\frac{25}{144}f\left(\frac{2}{5}\right)+\frac{25}{144}f\left(\frac{3}{5}\right)+\frac{25}{96}f\left(\frac{4}{5}\right)+\frac{19}{288}f(1)$$
$$=0.9460830$$

该积分的准确值$I=0.9460831$。从例题的计算结果可知，$n=2$ 和$n=3$ 的结果比较接近；$n=4$ 和$n=5$ 的结果比较接近。因此，根据定理4.4，在实际应用时，常常用n 为偶数的Newton-Cotes 公式而不用n 为奇数的Newton-Cotes 公式。

4.4 复化求积方法

从Newton-Cotes 的截断误差公式中可以看出，当积分区间$[a,b]$较大时，低阶的Newton-Cotes 求积公式截断误差都比较大。由于高阶Newton-Cotes 求积公式是数值不稳定的，因此通过不断增加阶数来提高求积公式的精度是不可行的。但是，如果将积分区间$[a,b]$分成几个小区间(任意的)，在每个小区间上应用Newton-Cotes 求积公式，其截断误差必然会减小，然后再把每个小区间上的积分值累加起来，这样却能大大提高整个积分的精度。这是一种行之有效的方法，并称这种方法为复化求积方法。

常用的复化求积方法采用等分区间的做法，具体如下：

将区间$[a,b]$划分为n 等分，步长为$H=(b-a)/n$，分点$x_k=a+kH$，$k=0,1,2,\cdots,n$。先用低阶Newton-Cotes 求积公式求得每个子区间$[x_k,x_{k+1}]$上的积分值I_k，然后将它们累加起来求和，用$\sum_{k=0}^{n-1}I_k$作为所求积分$I=\int_a^b f(x)\mathrm{d}x$ 的近似值。

4.4.1 复化梯形公式

在区间$[a,b]$上采用复化求积方法，具体使用梯形求积公式进行计算，就得到复化梯形公式。用T_k 表示$f(x)$在子区间$[x_k,x_{k+1}]$上的积分值，T_n 表示$f(x)$在区间$[a,b]$上的积分值，有：

$$T_k=\frac{1}{2}H[f(x_k)+f(x_{k+1})]$$

其中

$$H=(b-a)/n,\quad x_k=a+kH\quad(k=0,1,2,\cdots,n)$$

故

$$T_n=\sum_{k=0}^{n-1}T_k=\frac{1}{2}H\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]$$

即：

$$T_n=\frac{1}{2}H\left[f(x_{k=0})+f(x_{k=n})+2\sum_{k=1}^{n-1}f(x_k)\right]\tag{4.19}$$

或

$$T_n=\frac{b-a}{2n}\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)\right]$$

截断误差用R_T 表示，由于$f(x)$在区间$[a,b]$上有连续的二阶导数，故有：

$$R_T=\sum_{k=0}^{n-1}-\frac{1}{12}H^3f^{(2)}(\eta_k)=-\frac{1}{12}H^3\sum_{k=0}^{n-1}f^{(2)}(\eta_k)\quad \eta_k\in[x_k,x_{k+1}]$$

即

$$R_T=-\frac{1}{12}H^3\cdot n\cdot f^{(2)}(\eta)=-\frac{(b-a)}{12}H^2f''(\eta)\quad \eta\in[a,b] \tag{4.20}$$

用MATLAB语言编写的复化梯形公式(4.19)的程序很简洁(程序名称为Trape.m)：

```
function I=Trape(f,h)
I=h*(sum(f)-(f(1)+f(length(f)))/2);
```

其调用格式为：

```
I=Trape(f,h)
```

其中，f是分点函数值的矩阵；h是区间的长度。

例4.6　分析用复化梯形法计算下述积分时，划分区间数n对误差的影响。

$$I=\int_0^2\sqrt{1+e^x}\,dx$$

已知积分精确值$I=4.006994$。

解　编写MATLAB计算程序如下：

```
clear; Iexact=4.006994;
a=0; b=2;
fprintf('\n Extended Trapezoidal Rule \n');
fprintf('\ n   no         I               Error \ n');
no=1;
for k=1 : 6
    no=2 * no;
    h=(b-a)/no; i=1 : no + 1;
    x=a + (i - 1) * h;          % 区间的各分点
    f=sqrt(1 + exp(x));         % 各分点的函数值
    I=trape(f,h);
    fprintf('     %3.0f     %10.5f     %10.5f \n',…
                 no,I,Iexact - I);
end
```

计算结果为：

```
Extended Trapezoidal Rule
No      I        Error
 2    4.08358   0.07659
 4    4.02619   0.01919
 8    4.01180   0.00480
16    4.00819   0.00120
32    4.00729   0.00030
64    4.00707   0.00008
```

在本程序中，对积分区间进行了6次划分(划分区间数加倍、步长减半)，进行了6次独立的计算。当n加倍时，误差大约减少为原来的1/4。

4.4.2　复化Simpson公式

在区间$[a,b]$上采用复化求积方法，具体使用Simpson求积公式进行计算，就得到复化

Simpson 公式。用S_k表示$f(x)$在子区间$[x_k,x_{k+1}]$上的积分值,S_n表示$f(x)$在区间$[a,b]$上的积分值,有:

$$S_k=\frac{1}{6}H[f(x_k)+4f(x_{k+1/2})+f(x_{k+1})]$$

式中,$x_{k+1/2}$为子区间$[x_k,x_{k+1}]$的中点,$H=(b-a)/n$。

$$S_n=\sum_{k=0}^{n-1}S_k=\frac{1}{6}H\sum_{k=0}^{n-1}[f(x_k)+4f(x_{k+1/2})+f(x_{k+1})]$$

故
$$S_n=\frac{1}{6}H\left[f(a)+4\sum_{k=0}^{n-1}f(x_{k+1/2})+2\sum_{k=1}^{n-1}f(x_k)+f(b)\right] \tag{4.20}$$

截断误差用R_S表示,由于$f(x)$在区间$[a,b]$上有连续的四阶导数,故有:

$$R_S=\sum_{k=0}^{n-1}-\frac{1}{2880}H^5f^{(4)}(\eta_k)=-\frac{1}{2880}H^5\sum_{k=0}^{n-1}f^{(4)}(\eta_k)\quad \eta_k\in[x_k,x_{k+1}]$$

故
$$R_S=-\frac{(b-a)}{2880}H^4f^{(4)}(\eta)=-\frac{(b-a)}{180}\left(\frac{H}{2}\right)^4f^{(4)}(\eta)\quad \eta\in[a,b] \tag{4.21}$$

在实际应用中,常常使用复化Simpson 方法。为了便于编程,可将4.20)式改写为:

$$S_n=\frac{1}{6}H\left\{f(a)-f(b)+\sum_{k=1}^{n}[4f(x_{k-1/2})+2f(x_k)]\right\} \tag{4.22}$$

4.4.3 复化 Cotes 公式

在区间$[a,b]$上采用复化求积方法,具体使用Cotes 求积公式进行计算,就得到复化Cotes公式为:

$$C_n=\frac{1}{90}H\left[7f(a)+32\sum_{k=0}^{n-1}f(x_{k+1/4})+12\sum_{k=0}^{n-1}f(x_{k+1/2})+32\sum_{k=0}^{n-1}f(x_{k+3/4})+14\sum_{k=1}^{n-1}f(x_k)+7f(b)\right]$$

截断误差为:

$$R_C=-\frac{2(b-a)}{945}\left(\frac{H}{4}\right)^6f^{(6)}(\eta)\quad \eta\in[a,b]$$

从复化求积的余项公式中可以看出,复化梯形公式、复化Simpson 公式、复化Cotes 公式的余项和步长的关系为$R_T=O(h^2)$,$R_S=O(h^4)$,$R_C=O(h^6)$。因此,当$H\to0$或$n\to\infty$时,T_n,S_n,$C_n\to I$。

例4.7 给出函数$f(x)$的数据如表4.3所示,试计算积分$I=\int_0^1f(x)\mathrm{d}x$。

表4.3 函数$f(x)$的已知数据

x	$f(x)=\sin x/x$	x	$f(x)=\sin x/x$
0	1	5/8	0.936155636704739
1/8	0.997397867081822	3/4	0.908851680031112
1/4	0.989615837018092	7/8	0.877192573984031
3/8	0.97672674422946	1	0.841470984807896
1/2	0.958851077208406		

解 用复化梯形公式计算(8等分区间)可得:

$$T_8=\frac{1}{2}\times\frac{1}{8}\left\{f(0)+2\left[f\left(\frac{1}{8}\right)+f\left(\frac{1}{4}\right)+f\left(\frac{3}{8}\right)+f\left(\frac{1}{2}\right)+f\left(\frac{5}{8}\right)+f\left(\frac{3}{4}\right)+f\left(\frac{7}{8}\right)\right]+f(1)\right\}$$

$=0.9456909$

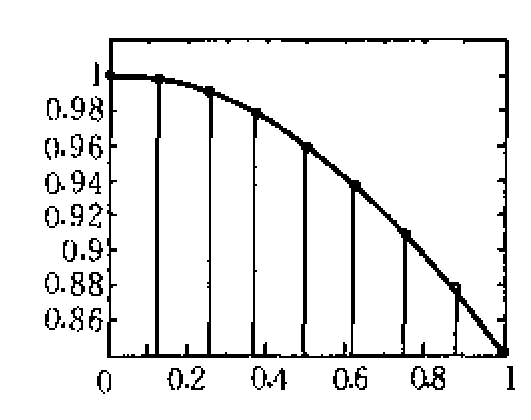
用复化 Simpson 公式计算(4 等分区间)可得：

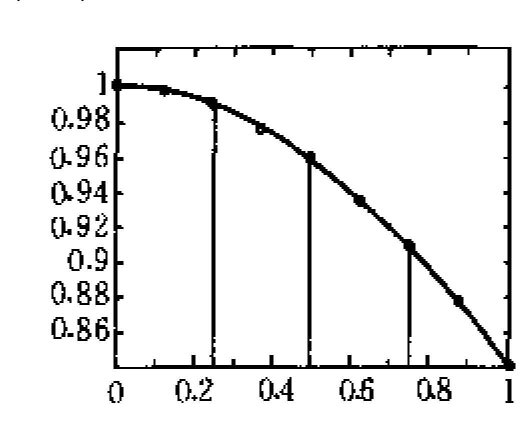
$$S_4=\frac{1}{6}\times\frac{1}{4}\left[f(0)+4f\left(\frac{1}{8}\right)+2f\left(\frac{1}{4}\right)+4f\left(\frac{3}{8}\right)+2f\left(\frac{1}{2}\right)+4f\left(\frac{5}{8}\right)+2f\left(\frac{3}{4}\right)+4f\left(\frac{7}{8}\right)+f(1)\right]$$

$$=0.9460832$$

用复化 Cotes 公式计算(2 等分区间)可得：

$$C_2=\frac{1}{90}\times\frac{1}{2}\left[7f(0)+32f\left(\frac{1}{8}\right)+12f\left(\frac{1}{4}\right)+32f\left(\frac{3}{8}\right)+14f\left(\frac{1}{2}\right)+32f\left(\frac{5}{8}\right)+12f\left(\frac{3}{4}\right)+32f\left(\frac{7}{8}\right)+7f(1)\right]=0.9460829$$





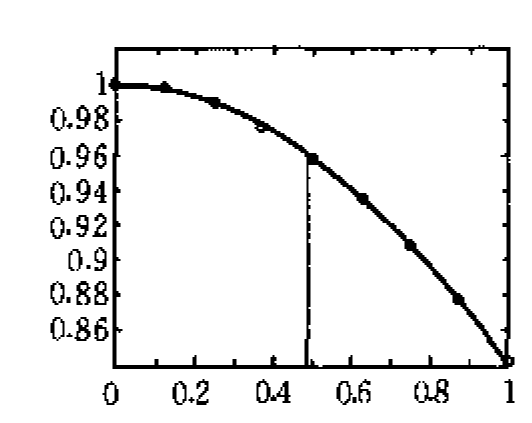


图 4.2 复化梯形法、复化 Simpson 法和复化 Cotes 法的图解

图4.2 给出了用三种计算方法计算的结果示意图。计算结果和精确值0.9460831 相比较，T_8 有2 位有效数字，S_4 有6 位有效数字，C_2 有5 位有效数字。由于复化Simpson 法的区间小，故复化Simpson 法比复化Cotes 法的计算精度高。因此，复化Simpson 法是一种常用的数值积分公式。

4.5 Romberg(龙贝格)积分法

4.5.1 变步长积分法

复化求积方法对提高积分精度是行之有效的，但必须事先给出恰当的步长h。如果步长太长则难以保证精度，太小则增加计算量。在实际应用中，一般采用变步长积分法来解决该问题，即让步长不断减小，考察在不同步长条件下的计算结果的精度，一旦计算结果满足精度要求，则可以停止计算，给出正确答案。这是面向计算机的解决方案。在人工计算的条件下，一般采用一种特殊的变步长法，即采用逐次二分积分区间的方法得到不断减半的步长，再应用复化求积方法，可推导出某种递推公式，既可减少计算量，又可以较快达到计算精度的要求。下面要介绍的正是这种变步长积分法。

1. 变步长梯形法

对于$I=\int_a^b f(x)\mathrm{d}x$，将积分区间$[a,b]$分成$n$ 等分，则共有$n+1$ 个分点：

$$x_k=a+kH\quad(k=0,1,2,\cdots,n),H=(b-a)/n$$

设此时用复化梯形公式求得的积分值为T_n。进一步地，将积分区间$[a,b]$分成$2n$ 等分，则

共有$2n+1$个分点,设此时用复化梯形公式求得的积分值为T_{2n}。显然,在计算T_{2n}的过程中所用的分点有一半是计算T_n时用过的,重复计算是浪费。为此,有必要研究T_n和T_{2n}之间的关系。

对于任意一个子区间$[x_k,x_{k+1}]$,用梯形公式计算的积分值记为$T_{1,k}$,在其中间增加一个节点$x_{k+1/2}=\frac{1}{2}(x_k+x_{k+1})$后,用复化梯形公式计算的积分值记为$T_{2,k}$。则有:

$$T_{1,k}=\frac{1}{2}H[f(x_k)+f(x_{k+1})]$$

$$T_{2,k}=\frac{1}{4}H[f(x_k)+2f(x_{k+1/2})+f(x_{k+1})]$$

分析$T_{2,k}$和$T_{1,k}$的关系,可得:

$$T_{2,k}=\frac{1}{2}T_{1,k}+\frac{1}{2}H\cdot f(x_{k+1/2})$$

将此式从$k=0,1,\cdots,n-1$累加求和,即得到用复化求积法得到的积分值为:

$$\sum_{k=0}^{n-1}T_{2,k}=\sum_{k=0}^{n-1}\frac{1}{2}T_{1,k}+\sum_{k=0}^{n-1}\frac{1}{2}H\cdot f(x_{k+1/2})$$

$$T_{2n}=\frac{1}{2}T_n+\frac{1}{2}H\sum_{k=0}^{n-1}f(x_{k+1/2}) \tag{4.23}$$

或

$$T_{2n}=\frac{1}{2}T_n+h\sum_{k=0}^{n-1}f(x_{k+1/2})$$

式中,$\sum_{k=0}^{n-1}f(x_{k+1/2})$为区间$[a,b]$分成$n$等分后再次二分而增加的新节点的函数值之和。要特别注意的是$H=(b-a)/n$为区间$[a,b]$分成n等分的步长,$h=\frac{b-a}{2n}$为区间$[a,b]$分成$2n$等分的步长,$x_{k+1/2}=a+\left(k+\frac{1}{2}\right)H$。

如果将区间$[a,b]$继续分成$4n$等分、$8n$等分、…、$2^in(i=0,1,\cdots)$等分,均可按照公式(4.23)递推计算出来。

在实际计算中,利用某二分前后两次积分值之差的绝对值$|T_{2n}-T_n|\leqslant\varepsilon$,来判断积分近似值$T_{2n}$是否满足精度要求。

例4.8　用变步长梯形法计算$\int_0^1\frac{\sin x}{x}dx$。

解　(1) 整个区间[0,1]的积分。当$h=1,H=1,f(0)=1,f(1)=0.841410$时,有:

$$T_1=\frac{1}{2}[f(0)+f(1)]=\frac{1}{2}(1+0.8414710)=0.9207355$$

(2) 第1次二分区间。当$h=1/2,H=1$,分点函数值$f(1/2)=0.9588510$时,有:

$$T_2=\frac{1}{2}T_1+\frac{1}{2}f\left(\frac{1}{2}\right)=\frac{1}{2}\times0.9207355+\frac{1}{2}\times0.9588510=0.9397933$$

(3) 第2次二分区间。$h=1/4,H=1/2$,计算新分点上的函数值$f(1/4)=0.9896158$,$f(3/4)=0.9088516$,则有:

$$T_4=\frac{1}{2}T_2+\frac{1}{4}\left[f\left(\frac{1}{2}\right)+f\left(\frac{3}{4}\right)\right]=0.9445135$$

(4) 第3次二分区间。当$h=1/8,H=1/4$,计算新分点上的函数值$f(1/8)=0.9973979$,$f(3/8)=0.9767267$,$f(5/8)=0.9361551$,$f(7/8)=0.8771926$时,则有:

$$T_8=\frac{1}{2}T_4+\frac{1}{8}\left[f\left(\frac{1}{8}\right)+f\left(\frac{3}{8}\right)+f\left(\frac{5}{8}\right)+f\left(\frac{7}{8}\right)\right]=0.9456909$$

这样，不断二分下去，二分10次以后即可得到精确值0.9460831。其计算结果如表4.4所示。

表4.4　例4.8的计算结果

k	T_n	k	T_n	k	T_n
0	0.9207355	4	0.9459850	8	0.9460827
1	0.9397933	5	0.9460596	9	0.9460830
2	0.9445135	6	0.9460769	10	0.9460831
3	0.9456909	7	0.9460815		

2. 变步长梯形法的误差

在变步长梯形法中，把区间$[a,b]$分成n等分后，用复化梯形公式计算积分I的近似值为T_n，截断误差为：

$$R_n=I-T_n=-\frac{(b-a)}{12}\left(\frac{b-a}{n}\right)^2 f^{(2)}(\eta_n)$$

把区间$[a,b]$分成$2n$等分后，用复化梯形公式计算积分I的近似值为T_{2n}，截断误差为：

$$R_{2n}=I-T_{2n}=-\frac{(b-a)}{12}\left(\frac{b-a}{2n}\right)^2 f^{(2)}(\eta_{2n})$$

当$f^{(2)}(\eta)$在区间$[a,b]$上变化不大时，有$f^{(2)}(\eta_n)\approx f^{(2)}(\eta_{2n})$，故

$$\frac{R_{2n}}{R_n}=\frac{I-T_{2n}}{I-T_n}\approx\frac{1}{4}$$

整理，得事后误差估计式为：

$$I-T_{2n}\approx\frac{1}{3}(T_{2n}-T_n)$$

也就是说，用前后两次计算的结果之差来估计误差，两者越接近则精度越高。这就是所谓的误差的事后估计。将I和T_{2n}的误差补偿给T_{2n}，得到比T_{2n}更精确的积分近似值：

$$\bar{I}=\frac{4}{3}T_{2n}-\frac{1}{3}T_n \tag{4.24}$$

将例4.8中所求得的T_4和T_8代入(4.24)式，则有$\bar{I}=0.9460833$，它与准确值0.9460831相比有6位有效数字。显然，精度大大提高，而且速度也加快了。

4.5.2　Romberg积分法

变步长梯形法方法简单，但精度低而且收敛速度慢。如果在保留这种方法使用的逐步二分思想的基础上，将用误差事后估计法所求得的误差作为积分近似值的补偿值，以便进一步提高精度，同时又引入加速收敛的技术，则就可以得到一种更加完善的数值积分方法，即Romberg积分法。

下面就具体讨论这种积分方法。即：将T_{2n}和T_n的表达式代入(4.24)式，得：

$$\bar{I}=\frac{4}{3}\left(\frac{1}{2}\cdot\frac{1}{2}H\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)\right]+\frac{1}{2}H\sum_{k=0}^{n-1}f(x_{k+1/2})\right)$$
$$-\frac{1}{3}\left(\frac{1}{2}H\left[f(a)+f(b)+2\sum_{k=1}^{n-1}f(x_k)\right]\right)$$

$$\bar{I}=\frac{1}{2}H\left(f(a)+4\sum_{k=0}^{n-1}f(x_{k+1/2})+2\sum_{k=1}^{n-1}f(x_k)+f(b)\right)=S_n$$

所以

$$S_n=\frac{4}{3}T_{2n}-\frac{1}{3}T_n \tag{4.25}$$

由此可得到结论一:用变步长梯形法二分前、后所求得的两个积分值T_n和T_{2n}进行线性组合,即可得到用复化Simpson 公式计算的积分值S_n。

进一步考察复化Simpson 公式。其截断误差与步长H 的4 次方成正比,即:

$$\frac{I-S_{2n}}{I-S_n}\cong\frac{1}{16}$$

整理得:

$$I-S_{2n}=\frac{1}{15}(S_{2n}-S_n)$$

同理有:

$$C_n=\frac{16}{15}S_{2n}-\frac{1}{15}S_n \tag{4.26}$$

由此得到结论二:用变步长Simpson 公式二分前、后所求得的两个积分值S_n和S_{2n}进行线性组合,即可得到用复化Cotes 公式计算的积分值C_n。

再考察复化Cotes 公式。其截断误差与步长H 的6 次方成正比,即:

$$\frac{I-C_{2n}}{I-C_n}\cong\frac{1}{64}$$

整理得:

$$R_n=\frac{64}{63}C_{2n}-\frac{1}{63}C_n \tag{4.27}$$

由此得到结论三:用变步长Cotes 公式二分前、后所求得的两个积分值C_n和C_{2n}进行线性组合,即可得到用Romberg 公式计算的积分值R_n,并称(4.27)式为Romberg 公式。

将变步长(区间数加倍、步长减半)梯形法的积分近似值通过线性组合得到变步长Simpson 法的积分近似值,再将变步长 Simpson 法的积分近似值通过线性组合得到变步长Cotes 法的积分近似值,再将变步长Cotes 法的积分近似值通过线性组合得到变步长Romberg 法的积分近似值。这种逐次二分积分区间并利用线性组合的加速方法将一个粗糙的积分值T_n序列逐步加工成精度较高的积分值R_n序列的方法,称为Romberg 积分法。

Romberg 积分法的具体计算步骤如下:

(1) 在区间$[a,b]$上,$h=(b-a)$,计算分点的函数值$f(a)$和$f(b)$后,用梯形求积公式计算T_1:

$$T_1=\frac{b-a}{2}[f(a)+f(b)]$$

(2) 将区间$[a,b]$二分,$h=(b-a)/2$,计算新增分点的函数值$f\left(\frac{a+b}{2}\right)$后,用(4.23)式计算$T_2$:

$$T_2=\frac{1}{2}T_1+h\sum_{k=0}^{n-1}f(x_{k+1/2})$$

用(4.25)式计算S_1:

$$S_1=\frac{4}{3}T_2-\frac{1}{3}T_1$$

（3）将区间$[a,b]$再二分，$h=(b-a)/4$，计算新增分点$f\left(a+\frac{a+b}{4}\right)$和$f\left(a+\frac{3}{4}(a+b)\right)$后，用(4.23)式计算$T_4$，用(4.25)式计算$S_2$，用(4.26)式计算$C_1$：

$$C_1=\frac{16}{15}S_2-\frac{1}{15}S_1$$

（4）将区间$[a,b]$再二分，$h=(b-a)/8$，计算新增分点的函数值后，用(4.23)式计算T_8，用(4.25)式计算S_4，用(4.26)式计算C_2，用(4.27)式计算R_1：

$$R_1=\frac{64}{63}C_2-\frac{1}{63}C_1$$

（5）将区间$[a,b]$再二分，$h=(b-a)/16$，计算新增分点的函数值后，计算T_{16}，S_8，C_4和R_2。

（6）将区间$[a,b]$继续二分下去，可得Romberg序列$R_1,R_2,R_4,R_8,\cdots,R_n,R_{2n}$直至满足精度要求$|R_{2n}-R_n|\leqslant\varepsilon$为止。

上述Romberg积分法的计算过程和结果可以用表4.5进行简洁表示。

表4.5　Romberg积分序列的计算

二分次数	区间等分数	梯形序列 代数精度1次	Simpson序列 代数精度3次	Cotes序列 代数精度5次	Romberg序列 代数精度7次
0	$1=2^0$	T_1			
1	$2=2^1$	T_2	S_1		
2	$4=2^2$	T_4	S_2	C_1	
3	$8=2^3$	T_8	S_4	C_2	R_1
4	$16=2^4$	T_{16}	S_8	C_4	R_2
⋮	⋮	⋮	⋮	⋮	⋮

例4.9　用Romberg方法计算积分$\int_0^1\frac{\sin x}{x}\mathrm{d}x$，要求得到积分准确值$I=0.9460831$。

解　(1) 区间$[0,1]$，$h=1$；$f(0)=1$，$f(1)=0.8417710$

$$T_1=\frac{1}{2}(1+0.8417710)=0.9207355$$

(2) 将$[0,1]$二分，$h=1/2$；$f\left(\frac{1}{2}\right)=0.9588510$

$$T_2=\frac{1}{2}\times0.9207355+\frac{1}{2}\times0.9588510=0.9397932$$

$$S_1=\frac{4}{3}T_2-\frac{1}{3}T_1=0.9460869$$

(3) 将$[0,1]$再二分，$h=1/4$；$f\left(\frac{1}{4}\right)=0.9896158$，$f\left(\frac{3}{4}\right)=0.9088516$

$$T_4=\frac{1}{2}T_2+\frac{1}{4}\left[f\left(\frac{1}{4}\right)+f\left(\frac{3}{4}\right)\right]=0.9445135$$

$$S_2=\frac{4}{3}T_4-\frac{1}{3}T_2=0.9460869$$

$$C_1=\frac{16}{15}S_2-\frac{1}{15}S_1=0.94608297$$

(4) 将[0,1]再二分,$h=1/8$;计算 4 个分点上的函数值 $f\left(\frac{1}{8}\right),f\left(\frac{3}{8}\right),f\left(\frac{5}{8}\right),f\left(\frac{7}{8}\right)$

$$T_8=\frac{1}{2}T_4+\frac{1}{8}\left[f\left(\frac{1}{8}\right)+f\left(\frac{3}{8}\right)+f\left(\frac{5}{8}\right)+f\left(\frac{7}{8}\right)\right]=0.9456909$$

$$S_4=\frac{4}{3}T_8-\frac{1}{3}T_4=0.94608337$$

$$C_2=\frac{16}{15}S_4-\frac{1}{15}S_2=0.94608313$$

$$R_1=\frac{14}{63}C_2-\frac{1}{63}C_1=0.9460831$$

由此可见,这里用二分 3 次($k=3$)的数据 T_2,T_4,T_8,通过 3 次加速获得了例 4.8 需要二分 10 次才能得到的结果。由于加速过程中不需要再计算函数值,所以加速过程的计算量非常小,可以忽略不计,而加速效果则是十分显著的。

4.6　Guass-Legendre(高斯 - 勒让德)求积方法

4.6.1　Guass 型求积公式

定义 4.2　在区间[a,b]内,如果由节点 $x_0,x_1,\cdots,x_n$ 构造的插值型求积公式

$$\int_a^b f(x)\mathrm{d}x\cong\sum_{k=0}^{n}A_kf(x_k)$$

具有 $2n+1$ 次代数精度,则称该求积公式为 Guass 型求积公式,求积节点 $x_k(k=0,1,\cdots,n)$ 为 Guass 点。

Guass 型求积公式是各种数值积分公式中精度较高的一种,它与梯形公式和 Simpson 公式等一样,也是插值型的。所不同者是,它所选择的 $n+1$ 个节点 $x_0,x_1,\cdots,x_n$ 并非等距节点,也取消了 x_0 和 x_n 与积分上下限 a 和 b 相重合的限制,其代数精度由此可提高到 $2n+1$ 次。

构造 Guass 型求积公式,首先要确定出 A_k 和 $x_k(k=0,1,\cdots,n)$ 两类系数,而求系数的关键点和难点在于求 Guass 点 x_k。下面以构造区间[-1,1]上的两点 Guass 公式

$$\int_{-1}^{1}f(x)\mathrm{d}x\cong A_0f(x_0)+A_1f(x_1)$$

为例,说明如何确定求积系数和求积节点。根据(4.9)式,列出非线性方程组为:

$$A_0+A_1=b-a=1-(-1)=2$$

$$A_0x_0+A_1x_1=\frac{b^2-a^2}{2}=0$$

$$A_0x_0^2+A_1x_1^2=\frac{b^3-a^3}{3}=\frac{2}{3}$$

$$A_0x_0^3+A_1x_1^3=\frac{b^4-a^4}{4}=0$$

解之,得:

$$A_0=A_1=1;\quad x_0=-\frac{1}{\sqrt{3}}\text{和}x_1=\frac{1}{\sqrt{3}}。$$

因此,两点 Guass 公式为:

$$\int_{-1}^{1} f(x)\mathrm{d}x \cong f\left(-\frac{1}{\sqrt{3}}\right)+f\left(\frac{1}{\sqrt{3}}\right)$$

若能用某种简便方法先求出求积节点x_k，非线性方程组(4.9)就变成线性方程组(4.8)，此时求积系数A_k就能够比较容易地求得。

定理4.5　求积节点$x_k(k=0,1,2,\cdots,n)$是Guass点的充要条件是，以这些点为零点的多项式

$$\Pi(x)=\prod_{k=0}^{n}(x-x_k)$$

与任意次数不超过n的多项式$p(x)$均正交，即：

$$\int_a^b p(x)\Pi(x)\mathrm{d}x=0$$

4.6.2　Legendre 多项式

Legendre 多项式由下列表达式定义：

$$\begin{cases} L_0(x)=1 \\ L_n(x)=\dfrac{1}{2^n n!}\dfrac{\mathrm{d}^n}{\mathrm{d}x^n}[(x^2-1)^n] \end{cases} \quad (n=1,2,\cdots)$$

从定义式可给出Legendre 多项式的前几项：

$$L_0(x)=1$$

$$L_1(x)=x \qquad L_2(x)=\frac{1}{2}(3x^2-1)$$

$$L_3(x)=\frac{1}{2}(5x^3-3x) \qquad L_4(x)=\frac{1}{8}(35x^4-20x^2+3)$$

Legendre 多项式的几个重要性质如下：

(1) 在区间[−1,1]上，n次Legendre 多项式$L_n(x)$与任意低于n次的多项式$p(x)$正交，即

$$\int_{-1}^{1} p(x)L_n(x)\mathrm{d}x=0$$

(2) Legendre 多项式所有的根在[−1,1]中，并且是不相同的实根。

(3) 递推关系为：

$$L_n(x)=\frac{2n-1}{n}xL_{n-1}(x)-\frac{n-1}{n}L_{n-2}(x) \quad n\geqslant 2$$

4.6.3　Guass-Legendre 求积公式

根据Legendre 多项式性质(1)，可以取Legendre 多项式的零点作为求积节点来构造Guass公式。这种求积方法就称为Guass-Legendre 求积法。

例如，为了构造3点Guass 公式

$$\int_{-1}^{1} f(x)\mathrm{d}x \cong \sum_{k=1}^{3} A_k f(x_k)$$

可取3次Legendre 多项式$L_3(x)$的零点

$$x_1=-\sqrt{3/5}, \quad x_2=0, \quad x_3=\sqrt{3/5}$$

作为求积节点。令求积公式对于$f(x)=1,x,x^2$都准确成立，则有：

$$\begin{cases} A_1+A_2+A_3=\int_{-1}^{1}\mathrm{d}x=2 \\ A_1x_1+A_2x_2+A_3x_3=\int_{-1}^{1}x\mathrm{d}x=0 \\ A_1x_1^2+A_2x_2^2+A_3x_3^2=\int_{-1}^{1}x^2\mathrm{d}x=2/3 \end{cases}$$

解之,得3个求积加权系数 $A_1=5/9, A_2=8/9, A_3=5/9$,最后得到3点Guass求积公式为:

$$\int_{-1}^{1}f(x)\mathrm{d}x\cong\frac{5}{9}f\left(-\sqrt{\frac{3}{5}}\right)+\frac{8}{9}f(0)+\frac{5}{9}f\left(\sqrt{\frac{3}{5}}\right)$$

如果区间 $[a,b]$ 是任意的,则需通过如下变换:

$$x=\frac{b-a}{2}t+\frac{a+b}{2}\quad \mathrm{d}x=\frac{b-a}{2}\mathrm{d}t$$

将在 $[a,b]$ 上的积分化为在区间 $[-1,1]$ 上的积分,即:

$$\int_a^b f(x)\mathrm{d}x\cong\frac{b-a}{2}\int_{-1}^{1}f\left(\frac{b-a}{2}t+\frac{a+b}{2}\right)\mathrm{d}t=\frac{b-a}{2}\int_{-1}^{1}\varphi(t)\mathrm{d}t \tag{4.28}$$

于是,在区间 $[a,b]$ 上的两点Guass-Legendre公式为:

$$\int_a^b f(x)\mathrm{d}x\cong\frac{b-a}{2}\left[f\left(-\frac{b-a}{2\sqrt{3}}+\frac{a+b}{2}\right)+f\left(\frac{b-a}{2\sqrt{3}}+\frac{b+a}{2}\right)\right]$$

例5.10 用两点Guass-Legendre公式计算 $\int_0^1\frac{\sin x}{x}\mathrm{d}x$。

解

$$\begin{aligned}\int_0^1\frac{\sin x}{x}\mathrm{d}x &\cong\frac{1}{2}\left[f\left(\frac{-1}{2\sqrt{3}}+\frac{1}{2}\right)+f\left(\frac{1}{2\sqrt{3}}+\frac{1}{2}\right)\right]\\ &=1/2[f(0.211401)+f(0.7885989)]\\ &=1/2(0.9925682+0.8995275)=0.9460478\end{aligned}$$

它有4位有效数字(精确值为0.9460831),比在同样条件下用变步长梯形法的结果(0.920735)精度提高了很多。

本 章 小 结

本章主要讨论数值积分的几种常用方法。这些方法可以用于对解析定义的或者以列表形式给出的函数的积分。其基本准则是:用多项式近似原函数,再用该多项式的积分近似原函数的积分。通过对节点横坐标做划分,可以导出许多不同的数值积分方法。牛顿-柯特斯公式要求等距划分,而高斯公式就不要求等距划分。牛顿-柯特斯公式中最简单的求积公式是梯形求积公式,该方法采用的是对被积函数的线性逼近。通过逐步二分方法可以逐步得到辛浦生求积公式—柯特斯求积公式—龙贝格求积公式。高斯求积公式的关键在于构造一个具有两个未知数(插值节点 x_k 和积分加权系数 A_k)的插值多项式。

本章应重点掌握的内容为:

(1) 机械求积法基本原理;

(2) 复化求积法和龙贝格变步长求积法基本原理;

(3) 各种数值积分方法的应用条件和计算精度;

(4) 高斯求积公式及其应用。

习 题 四

4.1 用复化梯形求积法计算下列积分(分别取$n=2,4,8,16$)：

(1) $\int_0^{\pi/4}\tan(x)\mathrm{d}x$ (2) $\int_0^1\exp(x)\mathrm{d}x$ (3) $\int_0^1\frac{1}{2+x}\mathrm{d}x$

4.2 用复化梯形法求下列积分(分别取$n=2,4,8,25,100$)，并给出数值积分结果与精确结果1之间的误差。

$$\int_0^{\pi/2}\sin(x)\mathrm{d}x$$

4.3 用复化梯形法计算如下积分(取$H=0.4,0.2,0.1$)：

$$\int_0^{0.8}f(x)\mathrm{d}x$$

其中，被积函数$f(x)$以下面的表格形式给出：

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8
$f(x)$	0	2.1220	3.0244	3.2568	3.1399	2.8579	2.5140	2.1639	1.8358

4.4 用4.3题的数据，并采用$H=0.2,0.1$的梯形法积分结果进行龙贝格积分，以求得更精确的积分值。

4.5 现给出表函数如下：

i	1	2	3	4	5
x_i	0	0.25	0.5	0.75	1.0
$f(x_i)$	0.9162	0.8109	0.6931	0.5596	0.4055

(1) 用复化梯形求积法计算下面积分(取$H=0.25,0.5$)：

$$I=\int_0^1 f(x)\mathrm{d}x$$

(2) 在(1)结果的基础上进行龙贝格积分，求得更精确的积分值。

4.6 导出下列三种矩形公式的余项：

(1) $\int_a^b f(x)\mathrm{d}x\approx(b-a)f(a)$

(2) $\int_a^b f(x)\mathrm{d}x\approx(b-a)f(b)$

(3) $\int_a^b f(x)\mathrm{d}x\approx(b-a)f\left(\frac{a+b}{2}\right)$

4.7 比较分别用梯形公式、两点高斯公式、辛普森公式和三点高斯公式计算下列积分的精度：

$$\int_3^1\sqrt{x-1.5}\,\mathrm{d}x$$

4.8 导出梯形公式$\int_a^b f(x)\mathrm{d}x\cong\left(\frac{b-a}{2}\right)[f(a)+f(b)]$的余项。

4.9 设定积分$\int_{0.5}^1\sqrt{x}\,\mathrm{d}x$，试用梯形公式、辛普森公式和柯特斯公式进行数值积分，并与精确值进行比较。

4.10 分别用下列方法计算积分$\int_0^1 e^x \mathrm{d}x$,要求准确到10^{-6}。

(1) 复化梯形法; (2) 复化辛普森法; (3) 龙贝格算法。

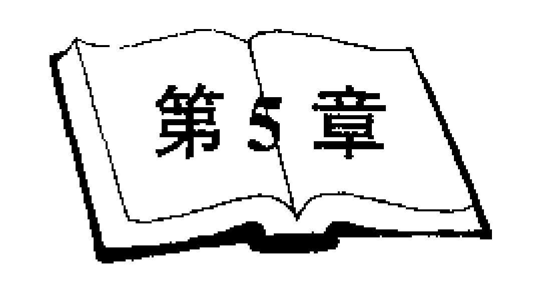
4.11 计算积分值:

(1) 用梯形公式、Simpson 公式分别计算$\int_0^1 3x^2 \mathrm{d}x$ 的值;

(2) 将区间[0,1]分成 2 等分后,再用梯形公式、Simpson 公式分别计算$\int_0^1 3x^2 \mathrm{d}x$ 的值;

(3) 将区间[0,1]分成 3,4,8 等分后,再用梯形公式计算$\int_0^1 3x^2 \mathrm{d}x$ 的值。

(4) 通过上述计算,给出你的结论。



第5章 常微分方程的数值解法

5.1 引　言

在工程实际问题中，数学模型通常会有各种表达形式，常微分方程就是其中常见的一种。这样，通过解微分方程就能得到实际问题数学模型的解。然而，用解析法一般只能求得某些类型微分方程的解而不能得到所有微分方程的解。例如

$$\begin{cases} y'(x) = e^{-x^2} & x\in[0,1] \\ y(0)=1 \end{cases}$$

等，是难以用解析法得到其解的。因此，应用数值方法解微分方程是很重要的方法。

本章主要讨论一阶常微分方程初值问题的数值解法，即对于要求解的函数 $y=y(x)$，已知其一阶导数和初始条件：

$$\begin{cases} y'(x) = f(x,y) & x\in[a,b] \\ y(x_0)=y_0 \end{cases} \tag{5.1}$$

研究如何使用数值方法得到该微分方程的解。

所谓微分方程初值问题的数值解法，就是将一个连续的微分方程初值问题转化为一个离散的差分方程初值问题，然后通过解差分方程而获得其数值解。即：对自变量的一系列离散节点 $x_k(k=0,1,2,\cdots,n,\cdots)$，计算出其精确解 $y(x_k)$ 的近似值 $y_k(k=1,2,\cdots,n,\cdots)$，(其中 y_0 是准确值)从而得到函数 $y=y(x)$ 的近似数值解 $y_k(k=1,2,\cdots,n,\cdots)$，如表5.1所示。

表5.1　函数 $y=y(x)$ 在离散节点的函数值

节点序号 k	0	1	2	…	n	…
离散节点 x_k	x_0	x_1	x_2	…	x_n	…
近似值 y_k	y_0	y_1	y_2	…	y_n	…
准确值 $y(x_k)$	$y(x_0)$	$y(x_1)$	$y(x_2)$	…	$y(x_n)$	…

注：表中 y_0 是准确值(初始值)。

关于微分方程解的存在性与惟一性，有下面的定理：

定理5.1　对于一阶常微分方程初值问题：

$$\begin{cases} y'(x) = f(x,y) & x\in[a,b] \\ y(x_0)=y_0 \end{cases}$$

如果 $f(x,y)$ 对 x 连续，且关于 y 满足Lipschitz(李普希兹)条件，即存在常数 $L>0$，使得

$$|f(x,y_1)-f(x,y_2)|\leqslant L|y_1-y_2|$$

对所有 $x\in[a,b]$ 以及任何实数 y_1,y_2 均成立，则初值问题在 $[a,b]$ 上有惟一解。

下面将依次讨论Euler方法、四阶Runge-Kutta方法以及一阶及高阶微分方程组的解法。

5.2 Euler(欧拉)方法

将微分方程(5.1)转化成差分方程，有几何方法、数值微分、数值积分和Taylor(泰勒)展开法等数值方法，这些方法将在推导微分方程的各种数值解法时得到具体运用。

5.2.1 显式Euler格式

从导数的定义可知，当步长$h=x_{n+1}-x_n\to 0$时，有：

$$y'(x)=\lim_{\Delta x\to 0}\frac{\Delta y}{\Delta x}=\lim_{\Delta x\to 0}\frac{f(x+\Delta x)-f(x)}{\Delta x}\cong\frac{y(x_{n+1})-y(x_n)}{x_{n+1}-x_n} \tag{5.2}$$

(5.2)式右边是导数的近似差商表示。因此，如果用向前差商、向后差商和中心差商分别代替微分方程(5.1)中的导数项$y'(x)$，则可相应地导出Euler方法的各种格式。

1. 显式Euler格式

若用在x_n点的向前差商$\dfrac{y(x_{n+1})-y(x_n)}{x_{n+1}-x_n}$代替微分方程(5.1)的导数项$y'(x_n)$，当$h=x_{n+1}-x_n$很小时，则有：

$$\frac{y(x_{n+1})-y(x_n)}{h}=f(x_n,y(x_n))$$

$$y(x_{n+1})=y(x_n)+hf(x_n,y(x_n))$$

若将$y(x_n)$的近似值y_n代入上式右端，并将计算结果记为y_{n+1}，则有：

$$y_{n+1}=y_n+hf(x_n,y_n)\quad(n=0,1,2,\cdots) \tag{5.3}$$

这样就得到了$y(x_{n+1})$近似值y_{n+1}的递推格式，即所谓的显式Euler格式。

若给出初始条件$y(x_0)=y_0$，则按此(5.3)式可逐步算出$y_1,y_2,\cdots$，即：

$$y_1=y_0+hy_0'=y_0+hf(x_0,y_0)$$

$$y_2=y_1+hf(x_1,y_1)$$

$$\cdots$$

$$y_n=y_{n-1}+hf(x_{n-1},y_{n-1})$$

这样就得到了函数$y=y(x)$在离散节点的函数近似值。

显式Euler格式也可以用Taylor展开法得到。

将函数$y=y(x)$在x_{n+1}点的值$y(x_{n+1})$展开为关于x_n点的Taylor级数，即：

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(\epsilon)\qquad x_n<\epsilon<x_{n+1}$$

略去余项，得：

$$y(x_{n+1})\cong y(x_n)+hy'(x_n)=y(x_n)+hf[x_n,y(x_n)]$$

$y(x_n)$用近似值y_n代替，就得到$y(x_{n+1})$的近似值y_{n+1}，即：

$$y_{n+1}=y_n+hf(x_n,y_n)$$

上式就是显式Euler格式。

2. 显式Euler格式的截断误差

为了确定显式Euler格式的计算误差，先引入微分方程数值解法的整体截断误差和局部

截断误差的概念。

定义5.1　在以某种数值方法求解满足Lipschitz条件的微分方程(5.1)时,如果产生惟一的逼近序列$\{y_n\}$,则称误差$E_n=y(x_n)-y_n(n=0,1,2,\cdots,N)$为该数值方法的整体截断误差。

定义5.2　在假设y_n是准确的即$y_n=y(x_n)$前提下,称用某种数值方法计算y_{n+1}的误差$e_{n+1}=y(x_{n+1})-y_{n+1}$为该数值方法在计算$y_{n+1}$时的局部截断误差。

定义5.3　若某种数值方法的局部截断误差为$O(h^{p+1})$,则称该数值方法的阶数为p。

显式Euler公式产生的局部截断误差可以用Taylor级数分析。

假设$y_n=y(x_n)$,将函数$y=y(x)$在x_{n+1}点的解析值$y(x_{n+1})$展开为关于x_n点的Taylor级数:

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2}y''(\varepsilon),\quad x_n<\varepsilon<x_{n+1}$$

又

$$y_{n+1}=y_n+hf(x_n,y_n)=y(x_n)+hf(x_n,y(x_n))=y(x_n)+hy'_n$$

两式相减,得:

$$y(x_{n+1})-y_{n+1}=\frac{h^2}{2}y''(\varepsilon)=O(h^2)$$

可见,显式Euler格式只有一阶精度,其局部截断误差与步长的平方成正比。虽然局部截断误差随着计算步距h的减小而减小,但是h过小并不利于计算精度,因为步距减小之后,计算次数会增加,必然导致舍入误差的增加。当舍入误差不可忽视时,计算就可能出现不稳定现象或者出现结果发散现象。

3. 显式Euler格式的几何意义

从几何上看,$y'(x)=f(x,y)$相当于在x,y平面上规定了一个方向场。以$y'=(x-y)/2$为例,作出其在矩形区域$R=\{(x,y):0\leqslant x\leqslant 5,0\leqslant y\leqslant 4\}$的方向场,如图5.1所示。如果$y(0)=1$,则$y(x)=3e^{-x/2}-2+x$;如果$y(0)=4$,则$y(x)=6e^{-x/2}-2+x$。从图中可以看出,曲线在每一点的切线和方向场的方向相同。

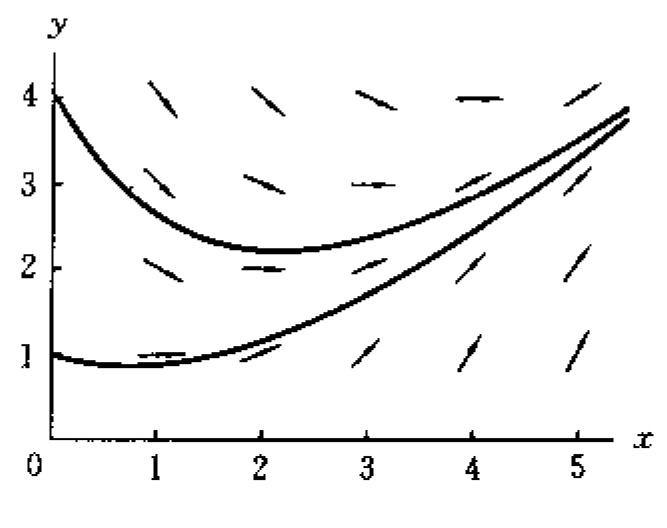


图5.1　微分方程的方向场

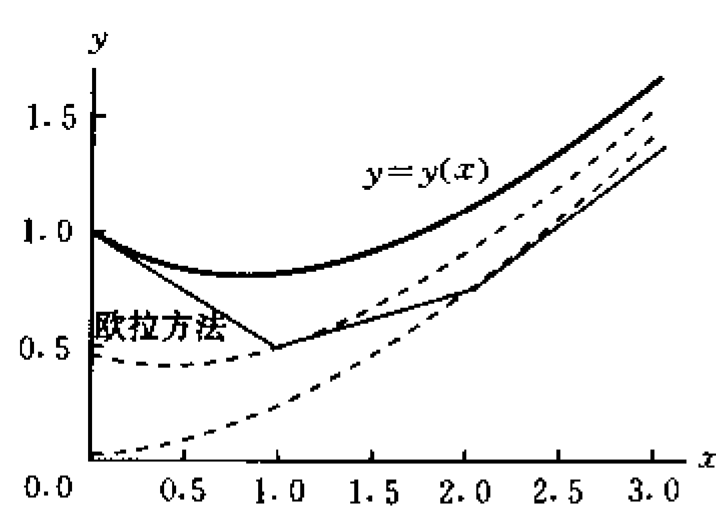


图5.2　Euler格式的几何意义

可见,从点(x_0,y_0)开始,用该点的斜率$f(x_0,y_0)$和步长h可以得到点(x_1,y_1),接着用点(x_1,y_1)的斜率$f(x_1,y_1)$和步长h可以得到点(x_2,y_2),照此做下去,可以得到更多的点(x_3,y_3),(x_4,y_4),…。这些点组成了一条折线,它与解曲线并不重合(点(x_0,y_0)出外),是解曲线的近似,如图5.2所示。

例5.1　用显式Euler格式求解初值问题:

$$\begin{cases}y'=y-\dfrac{2x}{y}, & 0<x<1\\ y(0)=1\end{cases}$$

解　对此问题显式Euler格式的具体形式为:

$$y_{n+1}=y_n+h\left(y_n-\frac{2x_n}{y_n}\right)$$

取$h=0.1$,计算结果列于表5.2中。

表5.2　例5.1题的求解结果

x_n	y_n	$y(x_n)$	x_n	y_n	$y(x_n)$
0.1	1.1000	1.0954	0.6	1.5090	1.4832
0.2	1.1918	1.1832	0.7	1.5803	1.5492
0.3	1.2774	1.2649	0.8	1.6498	1.6125
0.4	1.3582	1.3416	0.9	1.7178	1.6733
0.5	1.4351	1.4142	1.0	1.7848	1.7321

该微分方程的解析解为$y=\sqrt{1+2x}$。为了对比,将精确值$y(x_n)$也列入表5.2中。可以看出显式Euler格式精度比较低,其计算结果只有2位有效数字。

例5.2　设有质量为$M=70$kg的重物从高空垂直落下,假设初始时刻的垂直速度为0。空气阻力为$F_{air}=Cv^2$,其中,常数$C=0.27$kg/m,v为垂直速度,向下方向为正。试写出该重物的速度表达式,并画出$t\leqslant 20s$的解的图形。取积分步长$h=0.1s$。

解　由牛顿第二定理可知,力的平衡方程式如下:

$$M\frac{\mathrm{d}v(t)}{\mathrm{d}t}=-F_{air}+gM$$

可改写为:

$$\frac{\mathrm{d}v(t)}{\mathrm{d}t}=-\frac{C}{M}v^2+g\qquad v(0)=0$$

或

$$v'=f(v,t)=-\frac{C}{M}v^2+g\qquad v(0)=0$$

解上述微分方程即可得到速度表达式(写成显式Euler格式):

$$v_{n+1}=v_n+h\left(-\frac{C}{M}v_n^2+g\right)\qquad v(0)=0$$

其速度与时间的关系如图5.3所示,而求解该方程的MATLAB程序如下:

```
Clear,clf
C=0.27;M=70;
cm=C/M;
g=9.8;
tn=0;vn=0;
h=0.1;
n=0;
while tn<=20
    vn=vn+h*(-cm*vn.^2+g);
    tn=tn+h;
```

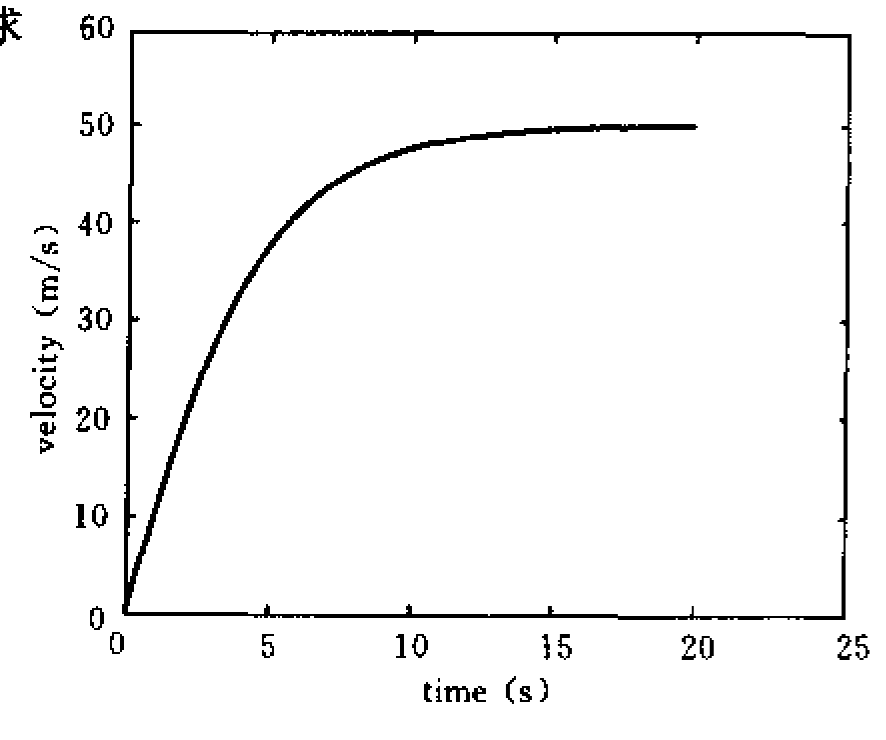


图5.3　落体速度和时间的关系

```
        n=n+1;
        T(n)=tn;
        V(n)=vn;
    end
    plot ([0,T],[0,V])
    xlabel ('time(s)')
    ylabel ('velocity (m/s)')
```

5.2.2 隐式Euler格式

1. 隐式Euler格式

若用在x_{n+1}点(不是在x_n点)的向后差商$\frac{y(x_{n+1})-y(x_n)}{x_{n+1}-x_n}$代替微分方程(5.1)的导数项$y'(x_{n+1})$,当$h=x_{n+1}-x_n$很小时,则有:

$$\frac{y(x_{n+1})-y(x_n)}{h}=f(x_{n+1},y(x_{n+1}))$$

这样可得:

$$y_{n+1}=y_n+hf(x_{n+1},y_{n+1}),\quad (n=0,1,2,\cdots) \tag{5.4}$$

由于等式的两端都同时有y_{n+1},故称为隐式Euler格式。

隐式Euler格式也只有一阶精度,但数值稳定性比显式Euler格式要好。通常采用迭代法(逐次代换法)来求解,使其逐步显式化。该方法的优点是:①绝对稳定;②如果解析解为正,则可以保证计算结果(数值解)也为正。

2. 预报-校正法

如果导函数为$f(x,y)=\varphi(x)+y$或$f(x,y)=y\cdot\varphi(x)$等形式时,则该微分方程的隐式Euler格式可以转化为显式Euler格式。即:

由 $$y_{n+1}=y_n+hf(x_{n+1},y_{n+1})=y_n+h[\varphi(x_{n+1})+y_{n+1}]$$

得 $$y_{n+1}=[y_n+h\varphi(x_{n+1})]/(1-h)$$

或 $$y_{n+1}=y_n+hf(x_{n+1},y_{n+1})=y_n+h[y_{n+1}\cdot\varphi(x_{n+1})]$$

得 $$y_{n+1}=y_n/[1-h\cdot\varphi(x_{n+1})]$$

但在一般情况下,要应用隐式Euler格式,应该采用预报-校正法。即:

先用显式Euler格式预报y_{n+1},得到近似值:

$$\bar{y}_{n+1}=y_n+hf(x_n,y_n)\quad (n=0,1,2,\cdots) \tag{5.5a}$$

再用隐式Euler格式校正$\bar{y}_{n+1}$,得到较为准确的值:

$$y_{n+1}=y_n+hf(x_{n+1},\bar{y}_{n+1})\quad (n=0,1,2,\cdots) \tag{5.5b}$$

(5.5)式称为预报-校正法公式。

5.2.3 两步Euler格式

为了提高精度,改用在x_n点的中心差商$[y(x_{n+1})-y(x_{n-1})]/(2h)$代替微分方程(5.1)即$y'(x_n)=f(x_n,y(x_n))$中的导数项,得:

$$y_{n+1}=y_{n-1}+2hf(x_n,y_n)\quad(n=1,2,\cdots)\tag{5.6}$$

这就是两步 Euler 格式。在计算 y_{n+1} 时,需要调用前两次的计算结果 y_{n-1} 和 y_n。

两步 Euler 格式的局部截断误差是:设 $y_n=y(x_n)$, $y_{n-1}=y(x_{n-1})$,则有:

$$y_{n+1}=y(x_{n-1})+2hf(x_n,y(x_n))=y(x_{n-1})+2hy'(x_n)$$

将 $y=y(x)$ 在 x_{n+1} 和 x_{n-1} 点的函数值 $y(x_{n+1})$ 和 $y(x_{n-1})$ 展开为关于 x_n 点的 Taylor 级数

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2!}y''(x_n)+\frac{h^3}{3!}y'''(\varepsilon_1)\qquad x_n<\varepsilon_1<x_{n+1}$$

$$y(x_{n-1})=y(x_n)-hy'(x_n)+\frac{h^2}{2!}y''(x_n)-\frac{h^3}{3!}y'''(\varepsilon_2)\qquad x_{n-1}<\varepsilon_2<x_n$$

两式相减,得:

$$y(x_{n+1})=y(x_{n-1})+2hy'(x_n)+\frac{h^3}{3}y'''(\varepsilon),\quad x_{n-1}<\varepsilon<x_{n+1}$$

故局部截断误差为:

$$y(x_{n+1})-y_{n+1}=\frac{h^3}{3}(\varepsilon)=O(h^3)$$

可见精度提高了,即两步 Euler 格式具有二阶精度。

5.2.4 改进的 Euler 格式

1. 梯形格式

现用数值积分的方法求解微分方程(5.1)。

将方程 $y'=f(x,y)$ 的两端从 x_n 到 x_{n+1} 积分,得:

$$\int_{x_n}^{x_{n+1}}y'\mathrm{d}x=\int_{x_n}^{x_{n+1}}f[x,y(x)]\mathrm{d}x$$

$$y(x_{n+1})-y(x_n)=\int_{x_n}^{x_{n+1}}f[x,y(x)]\mathrm{d}x\tag{5.7}$$

只要计算出(5.7)式中的积分项,就能得到 $y(x_{n+1})$。若采用数值积分的梯形公式,则有:

$$\int_{x_n}^{x_{n+1}}f[x,y(x)]\mathrm{d}x\cong\frac{1}{2}h[f(x_n,y(x_n))+f(x_{n+1},y(x_{n+1}))]$$

$$\cong\frac{1}{2}h[f(x_n,y_n)+f(x_{n+1},y_{n+1})]$$

将其代入(5.7)式,就得到 $y(x_{n+1})$ 的近似值 y_{n+1},即:

$$y_{n+1}=y_n+\frac{1}{2}h[f(x_n,y_n)+f(x_{n+1},y_{n+1})]\tag{5.8}$$

称(5.8)式为梯形格式。可以看出,梯形格式实际上是显式Euler 格式和隐式Euler 格式的算术平均值。要使用该公式,同样需要将公式进行显式化。

2. 改进的 Euler 格式

采用预报—校正方法,可将梯形公式显式化。即先用一个简单的格式求得一个初步的近似值,称为预报值;再用另一种格式进行校正,得到较为精确的校正值。如果用显式 Euler 格式进行预报,用隐式梯形格式进行校正,则可得:

预报 $\quad\bar{y}_{n+1}=y_n+hf(x_n,y_n)$

校正 $y_{n+1}=y_n+\frac{1}{2}h[f(x_n,y_n)+f(x_{n+1},\bar{y}_{n+1})]$

写成嵌套形式为：

$$y_{n+1}=y_n+\frac{1}{2}h[f(x_n,y_n)+f(x_{n+1},y_n+hf(x_n,y_n))]$$

为便于计算，可用平均化形式表示为：

$$\begin{cases}y_p=y_n+hf(x_n,y_n)\\ y_c=y_n+hf(x_{n+1},y_p)\\ y_{n+1}=\frac{1}{2}(y_p+y_c)\end{cases} \tag{5.9}$$

3. 改进的 Euler 格式的截断误差

改进的 Euler 格式的截断误差为 $O(h^3)$。

将(5.9)式改写为：

$$\begin{cases}y_{n+1}=y_n+\frac{1}{2}h(k_p+k_c)\\ k_p=f(x_n,y_n)\\ k_c=f(x_{n+1},y_n+hk_p)\end{cases} \tag{5.10}$$

其几何意义十分明显：用在 x_n 点的斜率 k_p 推算出 y_{n+1}，再计算出在 x_{n+1} 点的斜率 k_c，用 k_p 和 k_c 构成的平均斜率(可以减小单独使用 k_p 和 k_c 的偏差)代入 Euler 格式，最后计算出 y_{n+1}。即：

$$k_p=y'(x_n)$$

$$k_c=f(x_{n+1},y_n+hf(x_n,y_n))=f(x_n+h,y_n+hy'(x_n))$$

将 k_c 右端在点 (x_n,y_n) 处展开为 Taylor 级数，即：

$$k_c=f(x_n,y_n)+h\cdot f_x(x_n,y_n)+hy'(x_n)\cdot f_y(x_n,y_n)+O(h^2)$$

而 $$h\cdot f_x(x_n,y_n)+h\cdot y'(x_n)\cdot f_y(x_n,y_n)=h\cdot f'(x_n,y_n)=h\cdot y''(x_n)$$

故 $$k_c=f(x_n,y_n)+hy''(x_n)+O(h^2)=y'(x_n)+hy''(x_n)+O(h^2)$$

再将 k_p 和 k_c 代入 $y_{n+1}=y_n+h(k_p+k_c)/2$ 中，有：

$$y_{n+1}=y_n+\frac{1}{2}h(k_p+k_c)=y(x_n)+\frac{1}{2}h[2y'(x_n)+hy''(x_n)+O(h^2)]$$

$$y_{n+1}=y(x_n)+hy'(x_n)+\frac{1}{2}h^2y''(x_n)+\frac{1}{2}O(h^3)$$

将 $y(x_{n+1})$ 在点 (x_n,y_n) 处展开为 Taylor 级数，即：

$$y(x_{n+1})=y(x_n+h)=y(x_n)+hy'(x_n)+\frac{1}{2}h^2y''(x_n)+O(h^3)$$

所以 $$y(x_{n+1})-y_{n+1}=\frac{1}{2}O(h^3)$$

因此，改进的 Euler 格式的精度为 2 阶。

例 5.3 用改进的 Euler 格式求解初值问题

$$\begin{cases}y'=y-2x/y \qquad 0<x<1\\ y(0)=1\end{cases}$$

解 对此问题改进的 Euler 格式的具体形式为：

$$\begin{cases} y_p = y_n + h(y_n - 2x_n/y_n) \\ y_c = y_n + h(y_p - 2x_{n+1}/y_p) = y_n + h[y_p - 2(x_n + h)/y_p] \\ y_{n+1} = 1/2(y_p + y_c) \end{cases}$$

取 $h=0.1$,计算结果列于表 5.3 中(为了对比,精确值 $y(x_n)$ 也列入表中).

表 5.3 例 5.3 的计算结果

x_n	y_n	$y(x_n)$	x_n	y_n	$y(x_n)$
0.1	1.0959	1.0954	0.6	1.4860	1.4832
0.2	1.1841	1.1832	0.7	1.5525	1.5492
0.3	1.2662	1.2649	0.8	1.6165	1.6125
0.4	1.3434	1.3416	0.9	1.6782	1.6733
0.5	1.4164	1.4142	1.0	1.7397	1.7321

和显式 Euler 格式相比,精度明显地改善了,它具有 3 位有效数字。

5.3 Runge-Kutta(龙格 - 库塔)方法

由 5.2 节内容可知,Euler 方法有各种格式,但其精度最高不超过 2 阶,一般难以满足实际计算的精度要求。因此,有必要构造精度更高的数值计算公式求解微分方程。Runge-Kutta 方法就是一种高精度的经典的解常微分方程的单步方法。

5.3.1 Runge-Kutta 方法的基本思想

对于函数 $y=y(x)$,设 $y_n=y(x_n)$,将 $y(x_{n+1})$ 在点 x_n 处展开为 Taylor 级数:

$$y(x_{n+1})=y(x_n)+hy'(\varepsilon), \quad x_n<\varepsilon<x_{n+1}$$

利用 $y'(x_n)=f(x_n,y_n)$ 可得:

$$y(x_{n+1})=y(x_n)+hf(\varepsilon,y(\varepsilon)) \tag{5.11}$$

其中,$f(\varepsilon,y(\varepsilon))$ 为区间 (x_n,x_{n+1}) 上的平均斜率,记为 k^*。因此,只要能设法提供一种算法求得 k^*,就可相应地导出一种计算格式。

回顾一下前面所讲的 Euler 方法可知:

若取 x_n 点处的斜率 $k_1=f(x_n,y_n)$ 作为 k^*,则可得到显式 Euler 格式,但只有一阶精度;

若取 x_n 和 x_{n+1} 两点处的斜率 $k_1=f(x_n,y_n)$ 和 $k_2=f(x_{n+1},y_n+hk_1)$ 的算术平均值作为平均斜率 k^*,即 $k^*=(k_1+k_2)/2$,则得到改进的 Euler 格式,则有二阶精度。

由此推而广之,如果设法在区间 (x_n,x_{n+1}) 内取若干个点的斜率,然后将它们加权平均作为平均斜率 k^*,则可以构造一种具有更高精度的计算格式。这就是 Runge-Kutta 方法的基本思想。

5.3.2 二阶 Runge-Kutta 格式

在改进的 Euler 格式的基础上加以修改,即在区间 $[x_n,x_{n+1}]$ 内,除 x_n 外再取区间中任一点 p,则有:

$$x_{n+p}=x_n+ph \qquad 0<p\leqslant 1$$

取 x_n 和 x_{n+p} 两点处的斜率 k_1 和 k_2 加权平均作为平均斜率 k^*，即：

$$k^*=(1-\lambda)k_1+k_2 \qquad \lambda \text{ 为待定系数}$$

这样构造出的计算格式为：

$$y_{n+1}=y_n+h[(1-\lambda)k_1+\lambda k_2] \tag{5.12a}$$

其中：

$$k_1=f(x_n,y_n) \tag{5.12b}$$

$$k_2=f(x_{n+p},y_n+phk_1) \tag{5.12c}$$

(5.12)式称为二阶的Runge-Kutta格式，其中 p,λ 为待定参数。在确定 p,λ 的同时，使二阶Runge-Kutta格式具有较高的精度，具体推导如下：

根据假设 $y_n=y(x_n)$，有：

$$k_1=f(x_n,y_n)=y'(x_n)$$

$$k_2=f(x_{n+p},y_n+phk_1)=f(x_n+ph,y_n+phk_1)$$

将 k_2 右端在点 (x_n,y_n) 处展开为Taylor级数：

$$k_2=f(x_n,y_n)+ph\cdot f_x(x_n,y_n)+phy'(x_n)\cdot f_y(x_n,y_n)+O(h^2)$$

而 $$ph\cdot f_x(x_n,y_n)+ph\cdot y'(x_n)\cdot f_y(x_n,y_n)=ph\cdot f'(x_n,y_n)=ph\cdot y''(x_n)$$

故 $$k_2=f(x_n,y_n)+phy''(x_n)+O(h^2)=y'(x_n)+phy''(x_n)+O(h^2)$$

将 k_1 和 k_2 代入 $y_{n+1}=y_n+h[(1-\lambda)k_1+\lambda k_2]$ 中，有：

$$\begin{aligned}y_{n+1}&=y_n+h[(1-\lambda)k_1+\lambda k_2]\\&=y(x_n)+h[(1-\lambda)y'(x_n)+\lambda y'(x_n)+\lambda phy''(x_n)+\lambda O(h^2)]\end{aligned}$$

$$y_{n+1}=y(x_n)+hy'(x_n)+\lambda ph^2y''(x_n)+\lambda O(h^3)$$

再将 $y(x_{n+1})$ 在点 (x_n,y_n) 处展开为Taylor级数：

$$y(x_{n+1})=y(x_n+h)=y(x_n)+hy'(x_n)+\frac{1}{2}h^2y''(x_n)+O(h^3)$$

比较两式，得：

$$y(x_{n+1})-y_{n+1}=\left(\frac{1}{2}h^2-\lambda ph^2\right)y''(x_n)-\lambda O(h^3)$$

因此，当

$$\lambda p=1/2$$

时，二阶Runge-Kutta格式具有为2阶精度。可见二阶Runge-Kutta格式有很多具体形式：

若取 $\lambda=1/2,p=1$，则可得改进的Euler格式；

若取 $\lambda=3/4,p=2/3$，则可得Heun(休恩)格式；

若取 $\lambda=1,p=1/2$，则可得变形的Euler格式，即中点格式：

$$\begin{cases}y_{n+1}=y_n+hk_2\\k_1=f(x_n,y_n)\\k_2=f\left(x_{n+1/2},y_n+\dfrac{h}{2}k_1\right)\end{cases}$$

5.3.3 四阶Runge-Kutta格式

在区间 $[x_n,x_{n+1}]$ 内，使用两个不同的点可以构造出二阶Runge-Kutta格式。依此规律，在区间 $[x_n,x_{n+1}]$ 内，取3个不同的点可以构造出三阶Runge-Kutta格式；取4个不同的点可以构

造出四阶 Runge-Kutta 格式。对此可以加以证明。在实际中,应用最广泛的是四阶经典的 Runge-Kutta 格式:

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{6}(k_1+2k_2+2k_3+k_4) \\ k_1=f(x_n,y_n) \\ k_2=f\left(x_{n+1/2},y_n+\dfrac{h}{2}k_1\right) \\ k_3=f\left(x_{n+1/2},y_n+\dfrac{h}{2}k_2\right) \\ k_4=f(x_{n+1},y_n+hk_3) \end{cases} \tag{5.13}$$

四阶 Runge-Kutta 方法的优点如下:

(1) 它是一种高精度的单步法,可达四阶精度 $O(h^5)$;

(2) 数值稳定性较好;

(3) 只需知道一阶导数,无需明确定义或计算其他高阶导数;

(4) 只需给出 y_n 就能计算出 y_{n+1},所以能够自启动;

(5) 编程容易。由于有这些优点,所以应用非常广泛。

四阶 Runge-Kutta 法除经典的格式外,还有其他的格式。例如:

(1) Kutta 格式:

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{8}(k_1+3k_2+3k_3+k_4) \\ k_1=f(x_n,y_n) \\ k_2=f\left(x_{n+1/3},y_n+\dfrac{h}{3}k_1\right) \\ k_3=f\left(x_{n+2/3},y_n-\dfrac{h}{3}k_2+hk_2\right) \\ k_4=f(x_{n+1},y_n+hk_1-hk_2+hk_3) \end{cases}$$

(2) Gill 格式:

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{6}(k_1+(2-\sqrt{2})k_2+(2+\sqrt{2})k_3+k_4) \\ k_1=f(x_n,y_n) \\ k_2=f\left(x_{n+1/2},y_n+\dfrac{1}{2}hk_1\right) \\ k_3=f\left(x_{n+1/2},y_n+\dfrac{\sqrt{2}-1}{2}hk_1+\dfrac{2-\sqrt{2}}{2}hk_2\right) \\ k_4=f(x_{n+1},y_n-\dfrac{\sqrt{2}}{2}hk_2+\dfrac{2+\sqrt{2}}{2}hk_3) \end{cases}$$

例 5.4　取步长 $h=0.2$,用四阶 Runge-Kutta 格式求解下列初值问题:

$$\begin{cases} y'=y-2x/y \qquad 0<x<1 \\ y(0)=1 \end{cases}$$

解　对此问题,四阶 Runge-Kutta 格式具体形式为:

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{6}(k_1+2k_2+2k_3+k_4) \\ k_1=y_n-\dfrac{2x_n}{y_n} \\ k_2=\left(y_n+\dfrac{h}{2}k_1\right)-\dfrac{2(x_n+h/2)}{y_n+h/2\cdot k_1}=\left(y_n+\dfrac{h}{2}k_1\right)-\dfrac{2x_n+h}{y_n+h/2\cdot k_1} \\ k_3=\left(y_n+\dfrac{h}{2}k_2\right)-\dfrac{2(x_n+h/2)}{y_n+h/2\cdot k_2}=\left(y_n+\dfrac{h}{2}k_2\right)-\dfrac{2x_n+h}{y_n+h/2\cdot k_2} \\ k_4=(y_n+hk_3)-\dfrac{2(x_n+h)}{y_n+hk_3} \end{cases}$$

计算结果列于表5.4中(表中$y(x_n)$为精确值)。

表5.4　例5.4的计算结果

x_n	y_n	$y(x_n)$	x_n	y_n	$y(x_n)$
0.2	1.1832	1.1832	0.8	1.6125	1.6125
0.4	1.3417	1.3416	1.0	1.7321	1.7321
0.6	1.4833	1.4832			

与改进的Euler格式相比,其精度显然更高,它具有5位有效数字。虽然表面上看计算量较改进的Euler格式大1倍,但由于计算步长也加大了1倍,所以耗费的计算量几乎是一样的。

在MATLAB语言中,有专门解微分方程的功能函数,使用十分方便。采用四阶、五阶Runge-Kutta格式解微分方程的函数是ode45,其典型的调用格式为:

```
[T,Y]=ode45('F',TSPAN,Y0)
```

其中,F定义微分方程的形式$y'=F(t,y)$,TSPAN=[T0,TFINAL]表示积分区间从T0到TFINAL,Y0为初值,[T,Y]为计算结果的返回值。

例5.5　取步长$h=0.1$和$h=0.001$,用显式Euler格式、改进的Euler格式和四阶Runge-Kutta格式求解下列初值问题:

$$\begin{cases} y'=y-\dfrac{2x}{y} \qquad 0<x<1 \\ y(0)=1 \end{cases}$$

用Matlab语言编程计算,将计算结果绘成函数图形进行比较。

解　MATLAB程序如下:

```
clf
clear
% 显式Euler格式:
clear
x(1)=0;y(1)=1;
h=0.1;
n=0;
for j=1:50
    n=n+1;
    y(n+1)=y(n)+h*(y(n)-2*x(n)/y(n));
```

```
    x(n+1)=x(n)+h;
end
plot (x,y,'r')
xlabel ('x')
ylabel ('y=y(x)')
text (x(20),y(30),'comeuler1','FontSize',[8])
hold on
%  改进的Euler 格式:
clear
x(1)=0;y(1)=1;
h=0.1;
n=0;
for j=1:50
    n=n+1;
    yp=y(n)+h*(y(n)-2*x(n)/y(n));
    yc=y(n)+h*(yp-2*(x(n)+h)/yp);
    y(n+1)=(yc+yp)/2;
    x(n+1)=x(n)+h;
end
plot (x,y,'m')
text (x(42),y(50),'comeuler3','FontSize',[8])
hold on
%  四阶Runge-Kutta 格式:
clear
x(1)=0;y(1)=1;
h=0.1;
n=0;
for j=1:50
    n=n+1;
    k1=y(n)-2*x(n)/y(n);
    k2=y(n)+h*k1/2-2*(x(n)+h/2)/(y(n)+h*k1/2);
    k3=y(n)+h*k2/2-2*(x(n)+h/2)/(y(n)+h*k2/2);
    k4=y(n)+h*k3-2*(x(n)+h)/(y(n)+h*k3);
    y(n+1)=y(n)+h*(k1+2*k2+2*k3+k4)/6;
    x(n+1)=x(n)+h;
end
plot (x,y,'b')
text (x(45),y(50)+0.5,'comrt','FontSize',[8])
hold on
clear
```

```
x=0:0.5:5+1
y=sqrt(1+2*x);
plot (x,y,'o')
```

计算结果绘于图5.4和图5.5中。解析解以'o'的形式标示在图中。可见,减小步长有利于提高计算精度,但是,随着求解节点远离初始点,Euler格式计算的结果越来越偏离精确解,而四阶Runge-Kutta格式仍然具有较好的计算精度。

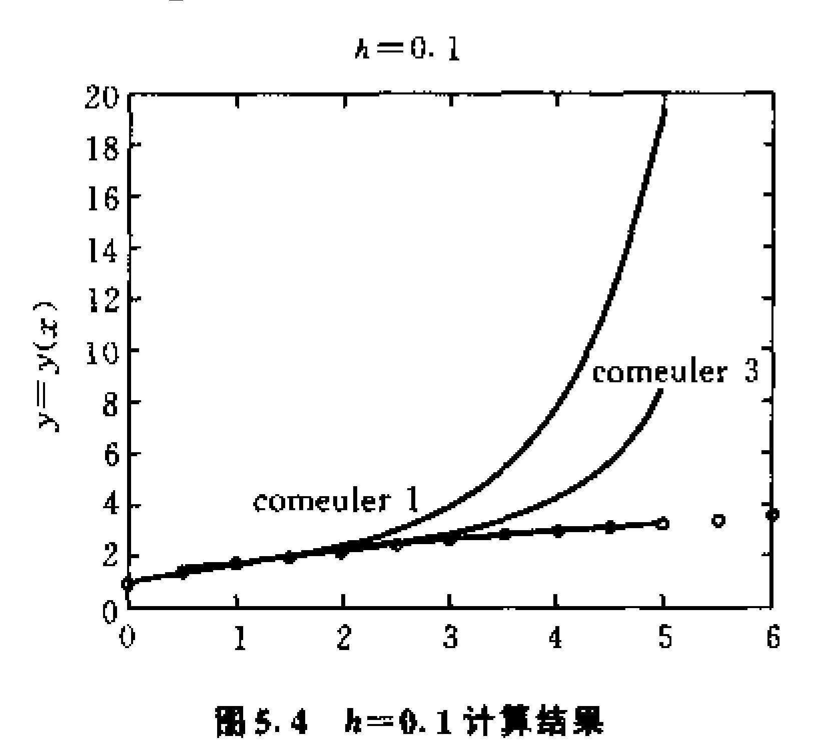


图5.4 $h=0.1$计算结果

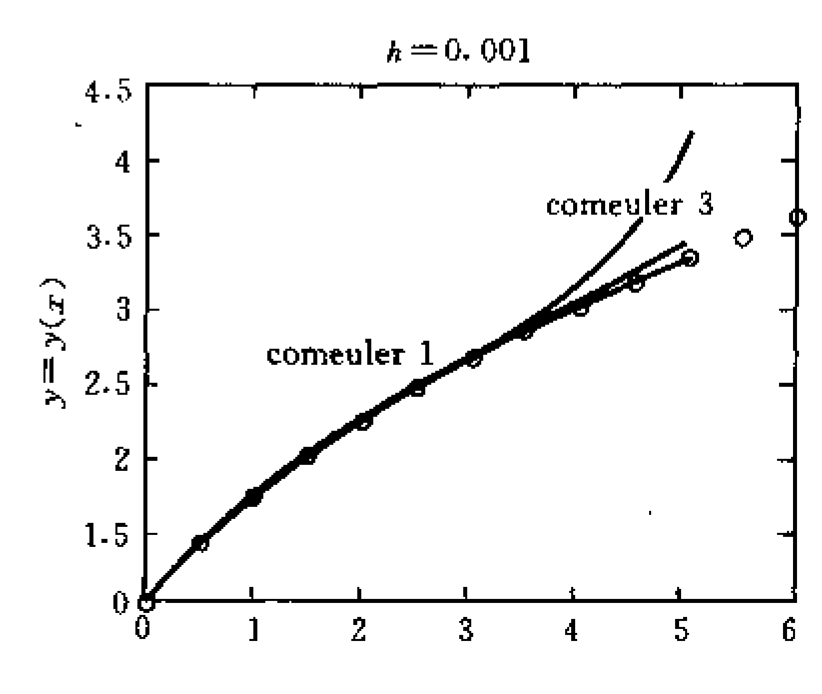


图5.5 $h=0.001$计算结果

需要说明的是,Runge-Kutta法是基于Taylor级数展开方法得到的。因此,要求所得到的解具有较好的光滑性,否则,用四阶Runge-Kutta格式所得的数值解,精度反而不如改进的Euler格式。使用时,应针对具体问题加以分析,选择恰当的算法。

5.3.4 变步长Runge-Kutta方法

在常微分方程的数值解法中存在一个恰当选取步长的问题。虽然Runge-Kutta方法的计算精度较高,可以允许选择较大的步长,但步长太大会导致解的不稳定;而步长太小又会增加计算量和导致舍入误差严重积累。因此,如果设计一种可以改变积分步长的算法,在函数值变化缓慢的区间采用较大步长,而在函数值变化快的区间采用较小步长,问题就能得到有效解决。下面以Runge-Kutta格式为例,描述变步长方法的算法。即:

(1) 从节点x_n出发,以h为步长,求得一个近似值,记为$y_{n+1}^{(h)}$;

(2) 将步长折半$h^*=h/2$,以步长h^*计算两步,又求得一个近似值,记为$y_{n+1}^{(h^*)}$;

(3) 计算$\Delta=|y_{n+1}^{(h)}-y_{n+1}^{(h^*)}|$;

(4) 对于给定的精度$0<\varepsilon_2<\varepsilon_1$:

如果$\Delta>\varepsilon_1$,则令$h=h^*$,重复步骤(1)、(2)和(3),直至$\Delta<\varepsilon_1$为止,并取最后一次得到的$y_{n+1}^{(h^*)}$作为计算结果,即$y_{n+1}=y_{n+1}^{(h^*)}$,步长为h;

如果$\varepsilon_2<\Delta<\varepsilon_1$,保持步长$h$不变,计算结果为$y_{n+1}=y_{n+1}^{(h)}$;

如果$\Delta<\varepsilon_2$,则从节点x_n出发,以步长h计算两次,求得的近似值记为$y_{n+1}^{(h)}$,令$h^*=2h$,从节点x_n出发,以h^*为步长,求得的近似值记为$y_{n+1}^{(h^*)}$,$\Delta=|y_{n+1}^{(h)}-y_{n+1}^{(h^*)}|$,直到$\Delta>\varepsilon_2$为止,取步长最后一次加倍前得到的$y_{n+1}^{(h)}$作为计算结果,即$y_{n+1}=y_{n+1}^{(h)}$,步长为$h$。

根据上述算法,不难作出程序框图。

由于Runge-Kutta方法的精度为$O(h^5)$,故有:

$$y(x_{n+1})-y_{n+1}^{(h)}\approx ch^5$$

将步长折半后，以步长 h^* 经过两步计算，得到 $y_{n+1}^{(h^*)}$，故两步的截断误差为：

$$y(x_{n+1})-y_{n+1}^{(h^*)}\approx 2c(h/2)^5$$

由此可见，步长折半后，截断误差为原来的1/16。其误差事后估计式为：

$$y(x_{n+1})-y_{n+1}^{(h^*)}\approx \frac{1}{15}(y_{n+1}^{(h^*)}-y_{n+1}^{(h)})$$

因此，在变步长法中，可取 $\varepsilon_2=\varepsilon_1/16$ 或更小。

5.4 单步法的收敛性与稳定性

5.4.1 单步法的收敛性

微分方程的数值解法是通过离散方法，将微分方程转化为差分方程而求解。由于各种Euler 格式和Runge-Kutta 格式都是单步法，故可以写成如下统一的格式：

$$\begin{cases}y_{n+1}=y_n+hF\{h,x_n,y_n\}\\ y(x_0)=y_0\end{cases}\tag{5.14}$$

用该差分方程求解微分方程，当 $h\to 0$ 时，差分方程的解 y_n 是否收敛于微分方程的精确解 $y(x_n)$呢？为讨论这一问题，假设当 $h\to 0$ 时，同时有 $n\to\infty$，并有下面的定义：

定义5.4 若单步法(5.14)对于固定的节点 $x_n=x_0+nh(n=1,2,\cdots)$，当 $h\to 0$(同时 $n\to\infty$)时，有

$$E_n=y(x_n)-y_n\to 0$$

则称该单步法是收敛的。

定理5.2 若初值问题(5.1)的单步法(5.14)的局部截断误差为 $O(h^{p+1})(p\geqslant 1)$，并且 $F\{h,x,y\}$对 y 满足Lipschitz 条件，即存在常数 $L>0$，使得

$$|F\{h,x,y\}-F\{h,x,\bar{y}\}|\leqslant L|y-\bar{y}|$$

对一切 $y,\bar{y}$ 均成立，则单步法(5.14)收敛，且有：

$$E_n=y(x_n)-y_n=O(h^p)$$

证明 根据局部误差的定义，有：

$$e_{n+1}=y(x_{n+1})-[y(x_n)+hF\{h,x_n,y(x_n)\}]=O(h^{p+1})$$

令
$$y_{n+1}^*=y(x_n)+hF\{h,x_n,y(x_n)\}$$

则有
$$|y(x_{n+1})-y_{n+1}^*|\leqslant ch^{p+1}$$

其中，常数 $c>0$。

因为
$$y_{n+1}=y_n+hF\{h,x_n,y_n\}$$

利用 F 关于 y 的Lipschitz 条件，有：

$$|y_{n+1}^*-y_{n+1}|\leqslant|y(x_n)-y_n|+h\cdot|F\{h,x_n,y(x_n)\}-F\{h,x_n,y_n\}|$$

即
$$|y_{n+1}^*-y_{n+1}|\leqslant|y(x_n)-y_n|+h\cdot L|y(x_n)-y_n|=(1+hL)|y(x_n)-y_n|$$

由于整体截断误差 $E_{n+1}=y(x_{n+1})-y_{n+1}$，所以

$$|E_{n+1}|=|y(x_{n+1})-y_{n+1}|\leqslant|y(x_{n+1})-y_{n+1}^*|+|y_{n+1}^*-y_{n+1}|\leqslant ch^{p+1}+(1+hL)|E_n|$$

如此递推下去，有：

$$|E_n| \leqslant ch^{p+1}[1+(1+hL)+\cdots+(1+hL)^{n-1}]+(1+hL)^n|E_0|$$

$$=ch^{p+1}\frac{(1+hL)^n-1}{(1+hL)-1}+(1+hL)^n|E_0|$$

因为 $y(x_0)=y_0$，故 $E_0=0$，又

$$0\leqslant 1+hL\leqslant 1+hL+\frac{(hL)^2}{2}e^{\xi}=e^{hL} \qquad 0\leqslant(1+hL)^n\leqslant e^{nhL}$$

所以

$$|E_n|\leqslant\frac{c}{L}h^p(e^{nhL}-1)+e^{nhL}|E_0|$$

当 $x=x_n=x_0+nh$ 固定时，e^{nhL}是常数，故

$$E_n=y(x_n)-y_n=O(h^p)$$

定理5.2给出了整体截断误差和局部截断误差之间的关系。由各种Euler格式的局部截断误差可知，当$f(x,y)$对y满足Lipschitz条件时，$h\to 0$，$E_n=y(x_n)-y_n\to 0$，即Euler格式是收敛的。

对经典的四阶Runge-Kutta格式

$$F\{h,x,y\}=\frac{1}{6}(k_1+2k_2+2k_3+k_4)$$

有

$$|k_1\{h,x,y\}-k_1\{h,x,\bar{y}\}|=|f(x,y)-f(x,\bar{y})|\leqslant L|y-\bar{y}|$$

$$|k_2\{h,x,y\}-k_2\{h,x,\bar{y}\}|=\left|f\left(x+\frac{h}{2},y+\frac{k_1\{h,x,y\}}{2}\right)-f\left(x+\frac{h}{2},\bar{y}+\frac{k_1\{h,x,\bar{y}\}}{2}\right)\right|$$

$$\leqslant L\left|\left(y+\frac{k_1\{h,x,y\}}{2}\right)-\left(\bar{y}+\frac{k_1\{h,x,\bar{y}\}}{2}\right)\right|$$

$$\leqslant L\left(1+\frac{1}{2}hL\right)|y-\bar{y}|$$

$$|k_3\{h,x,y\}-k_3\{h,x,\bar{y}\}|\leqslant L\left(1+\frac{1}{2}hL+\frac{1}{4}(hL)^2\right)|y-\bar{y}|$$

$$|k_4\{h,x,y\}-k_4\{h,x,\bar{y}\}|\leqslant L\left(1+\frac{1}{2}hL+\frac{1}{2}(hL)^2+\frac{1}{4}(hL)^3\right)|y-\bar{y}|$$

由 k_1,k_2,k_3,k_4 线性组合的$F\{h,x,y\}$也对y满足Lipschitz条件，即：

$$|F\{h,x,y\}-F\{h,x,\bar{y}\}|\leqslant L\left(1+\frac{1}{2}hL+\frac{1}{6}(hL)^2+\frac{1}{24}(hL)^3\right)|y-\bar{y}|=\bar{L}|y-\bar{y}|$$

由定理5.2可知，经典的四阶Runge-Kutta格式是收敛的。

5.4.2 单步法的稳定性

定义5.5 若初值问题(5.1)的单步法(5.14)对于任意给定的$\varepsilon>0$，存在$\delta>0$及$h_0>0$，当$0\leqslant h\leqslant h_0$时，差分方程(5.14)的实际数值运算公式

$$u_{n+1}=u_n+hF\{h,x_n,u_n\}+\eta_{n+1}$$

中的舍入误差η_{n+1}与初始误差y_0-u_0满足

$$|y_0-u_0|+\sum_{i=1}^{N}|\eta_i|\leqslant\delta$$

时，有

$$|y_i-u_i|\leqslant\varepsilon \qquad (i=1,2,\cdots,N),N=(b-a)/h$$

成立，则称单步法(5.14)是条件稳定的；若无h限制，则称为绝对稳定的。

定理 5.3　单步法(5.14)(假设 F 与 x_n 无关)是稳定的充要条件是差分方程(5.11)的特征根 r_i 满足 $|r_i|<1$。

由于稳定性问题比较复杂,故下面仅以 $y'=\lambda y(\lambda<0)$ 为例进行讨论。

该方程的显式 Euler 格式为:

$$y_{n+1}=y_n+\lambda h y_n$$

特征方程的根是 $\gamma=1+\lambda h$,当

$$|1+\lambda h|<1$$

时,显式 Euler 格式是条件稳定的,稳定条件是:

$$h\leqslant -\frac{2}{\lambda}$$

该方程的隐式 Euler 格式为:

$$y_{n+1}=y_n+\lambda h y_{n+1}$$

特征方程的根是 $\gamma=1/(1-\lambda h)$,由于 $\lambda<0$,所以

$$|\gamma|<1$$

故隐式 Euler 格式是绝对稳定的。

该方程经典的四阶 Runge-Kutta 格式为:

$$y_{n+1}=\left(1+\lambda h+\frac{1}{2}(\lambda h)^2+\frac{1}{6}(\lambda h)^3+\frac{1}{24}(\lambda h)^4\right)y_n$$

特征方程的根是:

$$\gamma=1+\lambda h1+\lambda h+\frac{1}{2}(\lambda h)^2+\frac{1}{6}(\lambda h)^3+\frac{1}{24}(\lambda h)^4$$

当 $|\gamma|<1$ 时,经典的四阶 Runge-Kutta 格式是条件稳定的,稳定条件是:

$$h\leqslant -\frac{2.78}{\lambda}$$

可见,显式公式均是条件稳定的,而绝对稳定的公式,一定是隐式的。

5.5　微分方程组与高阶方程的数值解法

5.5.1　一阶常微分方程组的数值解法

对于微分方程 $y'=f(x,y)$,若把 y 和 f 看作向量,则可推广到一阶微分方程组:

$$\begin{cases} y_i'=f_i(x,y_1,y_2,\cdots,y_m) \qquad (i=1,2,\cdots,m) \\ y_i(x_0)=y_{i0} \end{cases}$$

求解该一阶常微分方程组初值问题的四阶 Runge-Kutta 格式为:

$$\begin{cases} y_{i,n+1}=y_{i,n}+\dfrac{h}{6}(k_{i,1}+2k_{i,2}+2k_{i,3}+k_{i,4}) \qquad (i=1,2,\cdots,m) \\ k_{i,1}=f_i(x_n,y_{1,n},y_{2,n},\cdots,y_{m,n}) \\ k_{i,2}=f_i\left(x_{n+1/2},y_{1,n}+\dfrac{h}{2}k_{1,1},y_{2,n}+\dfrac{h}{2}k_{2,1},\cdots,y_{m,n}+\dfrac{h}{2}k_{m,1}\right) \\ k_{i,3}=f_i\left(x_{n+1/2},y_{1,n}+\dfrac{h}{2}k_{1,2},y_{2,n}+\dfrac{h}{2}k_{2,2},\cdots,y_{m,n}+\dfrac{h}{2}k_{m,2}\right) \\ k_{i,4}=f_i(x_n+h,y_{1,n}+hk_{1,3},y_{2,n}+hk_{2,3},\cdots,y_{m,n}+hk_{m,3}) \end{cases}$$

式中，$y_{i,n}$是第 i 个因变量 $y_i(x)$在节点 $x_n=x_0+nh$ 处的近似值。

如果 $m=2$，设 $y_1=y$，$y_2=z$，即：

$$\begin{cases} y'=f(x,y,z),y(x_0)=y_0 \\ z'=g(x,y,z),z(x_0)=z_0 \end{cases} \tag{5.15}$$

相应的四阶 Runge-Kutta 格式为：

$$\begin{cases} y_{n+1}=y_n+\dfrac{h}{6}(K_1+2K_2+2K_3+K_4) \\ z_{n+1}=z_n+\dfrac{h}{6}(L_1+2L_2+2L_3+L_4) \end{cases} \tag{5.16}$$

其中

$$K_1=f(x_n,y_n,z_n)$$
$$L_1=g(x_n,y_n,z_n)$$
$$K_2=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1,z_n+\frac{h}{2}L_1\right)$$
$$L_2=g\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_1,z_n+\frac{h}{2}L_1\right)$$
$$K_3=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_2,z_n+\frac{h}{2}L_2\right)$$
$$L_3=g\left(x_n+\frac{h}{2},y_n+\frac{h}{2}K_2,z_n+\frac{h}{2}L_2\right)$$
$$K_4=f(x_n+h,y_n+hK_3,z_n+hL_3)$$
$$L_4=g(x_n+h,y_n+hK_3,z_n+hL_3)$$

四阶 Runge-Kutta 格式是一步法（不用先预报再校正两步），利用节点处的值 y_n 和 z_n，按 $K_1,L_1\to K_2,L_2,\to K_3,L_3,\to K_4,L_4$ 顺序计算，然后按(5.16)式即可求得 y_{n+1}和 z_{n+1}。

一阶微分方程组(5.15)的显式 Euler 格式为：

$$\begin{cases} y_{n+1}=y_n+hf(x_n,y_n,z_n) \qquad y(x_0)=y_0 \\ z_{n+1}=z_n+hg(x_n,y_n,z_n) \qquad z(x_0)=z_0 \end{cases}$$

改进的 Euler 格式为：

预报
$$\begin{cases} \bar{y}_{n+1}=y_n+hf(x_n,y_n,z_n),y(x_0)=y_0 \\ \bar{z}_{n+1}=z_n+hg(x_n,y_n,z_n),z(x_0)=z_0 \end{cases}$$

校正
$$\begin{cases} y_{n+1}=y_n+\dfrac{1}{2}h[f(x_n,y_n,z_n)+f(x_{n+1},\bar{y}_{n+1},\bar{z}_{n+1})] \quad y(x_0)=y_0 \\ z_{n+1}=z_n+\dfrac{1}{2}h[g(x_n,y_n,z_n)+g(x_{n+1},\bar{y}_{n+1},\bar{z}_{n+1})] \quad z(x_0)=z_0 \end{cases}$$

5.5.2　高阶微分方程的解法

高阶微分方程的初值问题，原则上均可以化为一阶微分方程组来求解。下面以二阶常微分方程的初值问题为例加以具体说明。即，对于

$$\begin{cases} y''=f(x,y,y') \\ y(x_0)=y_0,y'(x_0)=y'_0 \end{cases}$$

若令 $z=y'$，则 $z'=y''=f(x,y,z)$，于是得到关于 y,z 的一阶方程组为：

$$\begin{cases} y'=z, & y(x_0)=y_0 \\ z'=f(x,y,z), & z(x_0)=y'_0 \end{cases}$$

其中，$y'=g(x,y,z)=z$。用四阶Runge-Kutta格式求解该一阶方程组的初值问题。即：

$$\begin{cases} y_{n+1}=y_n+\frac{h}{6}(K_1+2K_2+2K_3+K_4) \\ z_{n+1}=z_n+\frac{h}{6}(L_1+2L_2+2L_3+L_4) \end{cases}$$

式中

$$\begin{aligned} &K_1=z_n & &L_1=f(x_n,y_n,z_n) \\ &K_2=z_n+\frac{h}{2}L_1 & &L_2=f\left(x_{n+1/2},y_n+\frac{h}{2}K_1,z_n+\frac{h}{2}L_1\right) \\ &K_3=z_n+\frac{h}{2}L_2 & &L_3=f\left(x_{n+1/2},y_n+\frac{h}{2}K_2,z_n+\frac{h}{2}L_2\right) \\ &K_4=z_n+\frac{h}{2}L_3 & &L_4=f(x_{n+1},y_n+hK_3,z_n+hL_3) \end{aligned}$$

由于$y'=g(x,y,z)=z$是一个具体的函数，故可消去K_1,K_2,K_3,K_4，最后得：

$$\begin{cases} y_{n+1}=y_n+hz_n+\frac{h^2}{6}(L_1+L_2+L_3) \\ z_{n+1}=z_n+\frac{h}{6}(L_1+2L_2+2L_3+L_4) \end{cases}$$

其中

$$\begin{aligned} &L_1=f(x_n,y_n,z_n) \\ &L_2=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}z_n,z_n+\frac{h}{2}L_1\right) \\ &L_3=f\left(x_n+\frac{h}{2},y_n+\frac{h}{2}z_n+\frac{h^2}{4}L_1,z_n+\frac{h}{2}L_2\right) \\ &L_4=f\left(x_n+h,y_n+hz_n+\frac{h^2}{2}L_2,z_n+hL_3\right) \end{aligned}$$

例5.6　质量$M=0.5$kg的重物挂在弹簧振子(不计自重)下端，弹簧上端固定。重物受到弹簧振子的力为$R=-B\mathrm{d}x/\mathrm{d}t$，$B$为阻尼系数，$x$是重物离开固定点的位移。其运动方程为：

$$Mx''+Bx'+kx=0, x(0)=1, x'(0)=0 \tag{A}$$

式中，$x'=v$为重物的运动速度，k是弹簧系数(100N/m)，$B=10$Ns/m。试用四阶Runge-Kutta格式计算$x(t)$，$0<t\leqslant 0.05$s，积分步长$h=0.025$s。

解　题中的方程(A)可以写为：

$x'=f\equiv v, x(0)=1$　　——这是速度方程

$v'=g\equiv-(B/M)v-(k/M)x, v(0)=0$　　——这是加速度方程

(1) 用计算器计算：

令$a=B/M=20$，$b=k/M=200$。将四阶Runge-Kutta格式具体化：

当$t=t_0=0$时，有：

$$x_0=x(0)=1,\quad v=x'(0)=0$$

当$t=t_1=0.025$s时，有：

$K_1=hf(x_0,v_0,t_0)=hv_0=0.025\times 0=0$

$L_1=hg(x_0,v_0,t_0)=h(-20v_0-200x_0)=0.025\times(-20\times(0)-200\times(1))=-5$

$K_2=hf(x_0+K_1/2,v_0+L_1/2,t_0+h/2)=h(v_0+L_1/2)=0.025\times(0-5/2)=0.0625$

$L_2 = hg(x_0+K_1/2, v_0+L_1/2, t_0+h/2) = h[-20(v_0+L_1/2)-200(x_0+K_1/2)]$

$= 0.025\times[-20\times(0-5/2)-200\times(1+0/2)] = -3.75$

$K_3 = hf(x_0+K_2/2, v_0+L_2/2, t_0+h/2)$

$= h(v_0+L_2/2) = 0.025\times(0-3.75/2) = 0.046875$

$L_3 = hg(x_0+K_2/2, v_0+L_2/2, t_0+h/2) = h[-20(v_0+L_2/2)-200(x_0+K_2/2)]$

$= 0.025\times[-20\times(0-3.75/2)-200\times(1+0.0625/2)] = -3.9062$

$K_4 = hf(x_0+K_3, v_0+L_3, t_0+h)$

$= h(v_0+L_3) = 0.025\times(0-3.9062) = 0.09765$

$L_4 = hg(x_0+K_3, v_0+L_3, t_0+h) = h[-20(v_0+L_3)-200(x_0+K_3)]$

$= 0.025\times[-20\times(0-3.9062)-200\times(1+0.046875)] = -2.8125$

$x_1 = x_0+(1/6)(0+2)(-0.0625)+2(-0.046875)+(-0.09765) = 0.947265$

$v_1 = v_0+(1/6)(-5+2(-3.75)+2(-3.9062)+(-2.8125)) = -3.8541$

当$t=t_2=0.05$s 时，有：

$K_1 = hf(x_1, v_1, t_1) = hv_1 = 0.025\times(-3.8541) = -0.096354$

$L_1 = hg(x_1, v_1, t_1) = h(-20v_1-200x_1)$

$= 0.025\times(-20\times(-3.8541)-200\times(0.947265)) = -2.8092$

$K_2 = hf(x_1+K_1/2, v_1+L_1/2, t_1+h/2)$

$= h(v_1+L_1/2) = 0.025\times(-3.8541-2.8092/2) = -0.131469$

$L_2 = hg(x_1+K_1/2, v_1+L_1/2, t_1+h/2) = h[-20(v_1+L_1/2)-200(x_1+K_1/2)]$

$= 0.025\times[-20\times(-3.8541-2.8092/2)-200\times(0.947265-0.096354/2)]$

$= -1.866054$

$K_3 = hf(x_1+K_2/2, v_1+L_2/2, t_1+h/2)$

$= h(v_1+L_2/2) = 0.025\times(-3.8541-1.8066054/2) = -0.11968$

$L_3 = hg(x_1+K_2/2, v_1+L_2/2, t_1+h/2) = h[-20(v_1+L_2/2)-200(x_1+K_2/2)]$

$= 0.025\times[-20\times(-3.85416-1.866054/2)-200\times(0.947265-0.131469/2)]$

$= -2.014058$

$K_4 = hf(x_1+K_3, v_1+L_3, t_1+h)$

$= h(v_1+L_3) = 0.025\times(-3.85416-2.014058) = -0.146706$

$x_2 = x_1+(1/6)(-0.0963541+2(-0.0131469)+2(-0.119679)+(-0.146705))$

$= 0.823039$

$v_2 = v_1+(1/6)(-2.809244+2(-1.866048)+2(-2.014058)+(-1.203816))$

$= -5.816375$

(2) 用MATLAB程序计算：

为了便于MATLAB计算，将方程改写成矩阵形式（为了简化，设$B=0$），即：

$$Y' = MY+S$$

其中

$$Y=\begin{bmatrix} x \\ v \end{bmatrix},\quad M=\begin{bmatrix} 0 & 1 \\ -b & -a \end{bmatrix},\quad S=\begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

MATLAB程序如下：

```
echo off
```

```
clear
clf
hold off
M=0.5;B=0;k=100;
a=B/M;b=k/M;
M=[0,1;-b,-a],
Y(:,1)=[1;0];t(1)=0; % initial condition
h=0.025;
n=0;
T=0;
while T<1
    n=n+1;
    k1=h*M*Y(:,n);
    k2=h*M*(Y(:,n)+k1/2);
    k3=h*M*(Y(:,n)+k2/2);
    k4=h*M*(Y(:,n)+k3);
    Y(:,n+1)=Y(:,n)+(k1+2*k2+2*k3+k4)/6;
    t(n+1)=t(n)+h;
    T=t(n+1);
end
t,Y,
plot (t,Y(1,:),'-',t,Y(2,:),':');
xlabel ('t(s)') ,ylabel ('Y and v ')
text (t(5),Y(1,5)+2.0,'Displacement,x','FontSize',[18])
text (t(7),Y(2,7),' Velocity,v ','FontSize',[18])
axis ([0,1,-15,15])
```

图5.6示出了本题计算结果的图形。

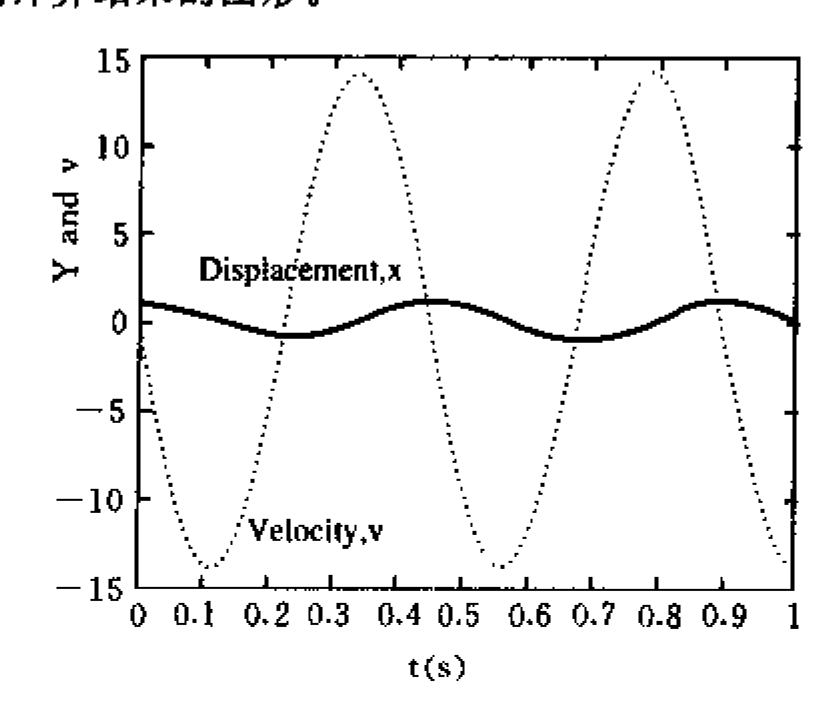


图5.6 例5.6计算结果的图形

本 章 小 结

常微分方程的数值解基于微分的差分近似原理。其最基本的方法是Euler法。根据差分形式的不同，有显式Euler(向前差商)格式、隐式Euler(向后差商)格式、二步Euler(中心差商)格式和改进的Euler格式。对Euler格式作些修改，可以得到Runge-Kutta格式，且最常用的是四阶经典Runge-Kutta格式。

本章主要掌握的内容是：

(1) 不同的Euler格式的构造原理，以及它们的格式结构；

(2) 计算精度的评价指标，以及不同的Euler格式的精度比较；

(3) 四阶Runge-Kutta格式的结构和计算流程以及计算精度；

(4) 微分方程组和高阶微分方程的数值解方法；

(5) 如何运用MATLAB的相关函数和程序解微分方程。

习 题 五

5.1　用向前Euler方法解初值问题：

$$y'=x^2+100y^2 \qquad y(0)=0$$

取步长$h=0.1$，计算到$x=0.3$(保留4位小数)。

5.2　用向前Euler方法手算下列问题(设$0\leqslant t\leqslant 0.5$，$h=0.1$)：

(1) $y'+ty=1, y(0)=1$

(2) $y'+3y=e^{-t}, y(0)=1$

(3) $y'=(t^2-y), y(0)=0.5$

5.3　用$h=0.5$的向前Euler方法计算初值问题：

$$y''(t)-0.05y'(t)+0.15y(t)=0 \qquad y'(0)=0, y(0)=1$$

并算出$y(1)$和$y(2)$。

5.4　用改进的Euler方法计算第5.2题，并写出MATLAB程序，绘出问题解的图形。

5.5　用改进的Euler法计算积分$y=\int_0^x e^{t^2}\mathrm{d}t$在$x=0.5, 0.75, 1.0$时的近似值。

5.6　用四阶Runge-Kutta方法计算5.2(1)题，并比较与Euler方法的计算精度。

5.7　用四阶Runge-Kutta方法求解下列方程：

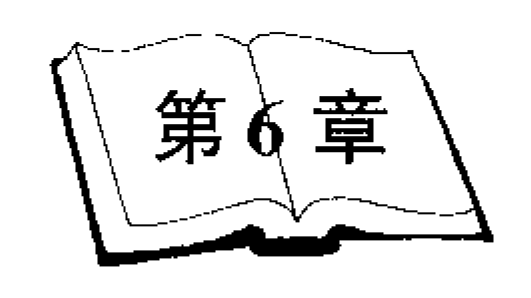
$$y'=-y/(t+y^2), \quad y(0)=1$$

取$h=1$，计算$y(1)$。

5.8　用四阶Runge-Kutta方法解初值问题：

$$\begin{cases} y''-2y^3=0 & 1<x<1.5 \\ y(1)=y'(1)=-1 \end{cases}$$

取$h=0.1$算到$x=1.5$，并与精确解$y=\dfrac{1}{x-2}$相比较。



第6章 线性方程组的数值解法

6.1 引　　言

一个n阶线性方程组

$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=b_1 \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=b_2 \\ \cdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=b_n \end{cases} \tag{6.1}$$

可以写成表达简洁的分量形式：

$$\sum_{j=1}^{n} a_{ij}x_j=b_i \qquad (i=1,2,\cdots,n) \tag{6.2}$$

也可写成矩阵形式：

$$AX=B \tag{6.3}$$

式中，系数矩阵$A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$，解向量$X=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$，常向量$B=\begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$

若系数矩阵A的行列式不等于零，即$\det A\neq 0$，则方程组$AX=B$有惟一解，并且可以用Cramer法则求出

$$x_i=\frac{\det A_i}{\det A} \qquad (i=1,2,3,\cdots,n)$$

其中，$\det A_i$表示将$\det A$的第i列元素全部换成常数项B的元素后所得到的n阶行列式。用Cramer法则求解方程组$AX=B$，总的乘法运算次数为$(n+1)(n-1)n!$。当n较大时，计算量大得惊人，即使利用现代先进和快速的计算机也难以完成计算工作。因此，线性方程组的数值解法常常使用直接法和迭代法两类方法。

6.2 解线性方程组的直接法

直接法的基本思想是，将结构上比较复杂的原始方程组，通过等价变换，转化成结构简单的方程组，然后再通过求解结构简单的方程组，就可以得到原始方程组的解。即：

$$AX=B \iff UX=Y$$

U矩阵通常是对角矩阵、三角矩阵或一些结构简单的矩阵。若计算过程中没有舍入误差，则直接法通过有限次的算术运算，可以求出方程组的精确解。

6.2.1　Guass(高斯)消去法

Guass 消去法也称顺序消去法,是一种求解线性方程组的直接方法,主要计算过程分为消元过程和回代过程两个步骤。

1. 消元过程

一般地,对 n 阶线性方程组

$$\begin{cases} a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+\cdots+a_{1j}^{(0)}x_j+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)} \\ a_{21}^{(0)}x_1+a_{22}^{(0)}x_2+\cdots+a_{2j}^{(0)}x_j+\cdots+a_{2n}^{(0)}x_n=b_2^{(0)} \\ \quad\vdots \\ a_{i1}^{(0)}x_1+a_{i2}^{(0)}x_2+\cdots+a_{ij}^{(0)}x_j+\cdots+a_{in}^{(0)}x_n=b_i^{(0)} \\ \quad\vdots \\ a_{n1}^{(0)}x_1+a_{n2}^{(0)}x_2+\cdots+a_{nj}^{(0)}x_j+\cdots+a_{nn}^{(0)}x_n=b_n^{(0)} \end{cases} \tag{6.4}$$

可以按如下方法进行消元:首先进行方程组第1列消元。假设 $a_{11}^{(0)}\neq0$,由方程组(6.4)的第1个方程,两边同时除以 $a_{11}^{(0)}$,得:

$$x_1+a_{12}^{(0)}/a_{11}^{(0)}x_2+\cdots+a_{1j}^{(0)}/a_{11}^{(0)}x_j+\cdots+a_{1n}^{(0)}/a_{11}^{(0)}x_n=b_1^{(0)}/a_{11}^{(0)} \tag{6.5}$$

再以方程组第 i 个(其中 $i>1$)方程变量 x_1 的系数 $a_{i1}^{(0)}$ 同时乘(6.5)式两边,得:

$$a_{i1}^{(0)}x_1+a_{12}^{(0)}a_{i1}^{(0)}/a_{11}^{(0)}x_2+\cdots+a_{1j}^{(0)}a_{i1}^{(0)}/a_{11}^{(0)}x_j+\cdots+a_{1n}^{(0)}a_{i1}^{(0)}/a_{11}^{(0)}x_n=b_1^{(0)}a_{i1}^{(0)}/a_{11}^{(0)} \tag{6.6}$$

用(6.6)式即可消去方程组第 i 个方程中的变量 x_1,而该方程其他 x_i 的系数和 b_i 则变为:

$$a_{ij}^{(1)}=a_{ij}^{(0)}-a_{1j}^{(0)}\frac{a_{i1}^{(0)}}{a_{11}^{(0)}}\quad(i=2,3,\cdots,n;j=1,2,3,\cdots,n)$$

$$b_i^{(1)}=b_i^{(0)}-b_1^{(0)}\frac{a_{i1}^{(0)}}{a_{11}^{(0)}}\quad(i=2,3,\cdots,n)$$

$i=2,3,\cdots,n$,这样可以消去方程组(6.4)第1个方程以下各个方程的变量 x_1,得到一个与方程组(6.4)等价的新方程组:

$$\begin{cases} a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+\cdots+a_{1j}^{(0)}x_j+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)} \\ \qquad a_{22}^{(1)}x_2+\cdots+a_{2j}^{(1)}x_j+\cdots+a_{2n}^{(1)}x_n=b_2^{(1)} \\ \qquad\quad\vdots \\ \qquad a_{i2}^{(1)}x_2+\cdots+a_{ij}^{(1)}x_j+\cdots+a_{in}^{(1)}x_n=b_i^{(1)} \\ \qquad\quad\vdots \\ \qquad a_{n2}^{(1)}x_2+\cdots+a_{nj}^{(1)}x_j+\cdots+a_{nn}^{(1)}x_n=b_n^{(1)} \end{cases} \tag{6.7}$$

接下来进行方程组第2列消元。假设 $a_{22}^{(1)}\neq0$,按照第1列消元的方法消去方程组(6.7)第2个方程以下各个方程的变量 x_2 后,又得到一个与方程组(6.7)等价的新方程组:

$$\begin{cases} a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+a_{13}^{(0)}x_3+\cdots+a_{1j}^{(0)}x_j+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)} \\ \qquad a_{22}^{(1)}x_2+a_{23}^{(1)}x_3+\cdots+a_{2j}^{(1)}x_j+\cdots+a_{2n}^{(1)}x_n=b_2^{(1)} \\ \qquad\quad\vdots \\ \qquad\qquad a_{i3}^{(2)}x_3+\cdots+a_{ij}^{(2)}x_j+\cdots+a_{in}^{(2)}x_n=b_i^{(2)} \\ \qquad\quad\vdots \\ \qquad\qquad a_{n3}^{(2)}x_3+\cdots+a_{nj}^{(2)}x_j+\cdots+a_{nn}^{(2)}x_n=b_n^{(2)} \end{cases}$$

其中，第i个(其中$i>2$)方程变量x_i的系数和b_i为：

$$a_{ij}^{(2)}=a_{ij}^{(1)}-a_{2j}^{(1)}\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}\quad(i=3,4,\cdots,n;j=2,3,4,\cdots,n)$$

$$b_i^{(2)}=b_i^{(1)}-b_2^{(1)}\frac{a_{i2}^{(1)}}{a_{22}^{(1)}}\quad(i=3,4,\cdots,n)$$

一般地，经过k次消元后(假设$a_{kk}^{(k-1)}\neq0$)，方程组第i个方程变量x_i的系数和b_i为

$$a_{ij}^{(k)}=a_{ij}^{(k-1)}-a_{kj}^{(k-1)}\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}\quad(k=1,2,\cdots,n-1;i=k+1,k+2,\cdots,n;j=k,k+1,\cdots,n)$$

$$b_i^{(k)}=b_i^{(k-1)}-b_k^{(k-1)}\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$$

最后，方程组经过$n-1$次消元后(假设$a_{nn}^{(n-1)}\neq0$)，得：

$$\begin{cases}a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+a_{13}^{(0)}x_3+\cdots+a_{1j}^{(0)}x_j+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)}\\ \qquad a_{22}^{(1)}x_2+a_{23}^{(1)}x_3+\cdots+a_{2j}^{(1)}x_j+\cdots+a_{2n}^{(1)}x_n=b_2^{(1)}\\ \qquad\vdots\\ \qquad a_{ii}^{(i-1)}x_i+\cdots+a_{in}^{(i-1)}x_n=b_i^{(i-1)}\\ \qquad\vdots\\ \qquad a_{nn}^{(n-1)}x_n=b_n^{(n-1)}\end{cases}\tag{6.8}$$

至此，方程组的消元过程全部完成。

2. 回代过程

从方程组(6.8)的最后一个方程出发解出x_n，然后逐步回代求出所有的x_i。即：

$$\begin{aligned}&x_n=b_n^{(n-1)}/a_{nn}^{(n-1)}\\&x_i=\left(b_i^{(i-1)}-\sum_{j=i+1}^{n}a_{ij}^{(i-1)}x_j\right)\Big/a_{ii}^{(i-1)}\quad(i=n-1,n-2,\cdots,1)\end{aligned}\tag{6.9}$$

可以看出，将一个结构复杂的方程组(6.4)经过消元后，可以变换成一个结构简单的方程组(6.8)，并且很容易求解出方程组的解。从方程组(6.4)变换成方程组(6.8)的过程，称为消元过程，根据(6.9)式依次求出x_i的过程($i=n,n-1,n-2,\cdots,2,1$)，称为回代过程。

例 6.1　用Guass消去法求解下列线性方程组：

$$\begin{cases}2x_1-x_2+3x_3=1\\4x_1+2x_2+5x_3=4\\x_1+2x_2\qquad=7\end{cases}\tag{A}$$

解　将方程组(A)中的第1个方程x_1的系数化为1，并消去其他方程中的x_1，得：

$$\begin{cases}x_1-0.5x_2+1.5x_3=0.5\\\qquad 4x_2-x_3=2\\\qquad 2.5x_2-1.5x_3=6.5\end{cases}\tag{B}$$

方程组(B)中的第1个方程保持不变，再使第2个方程x_2的系数化为1，并消去(B)中第3个方程里的x_2，得：

$$\begin{cases}x_1-0.5x_2+1.5x_3=0.5\\\qquad x_2-0.25x_3=0.5\\\qquad\qquad x_3=-6\end{cases}\tag{C}$$

以上两步完成消元过程。对方程组(C)从下至上依次回代，即得到方程组的解为：

$$x_3=-6,\quad x_2=-1,\quad x_1=9$$

在Guass消元过程中，如果把对角线上方的未知量的系数均化为零，使方程组的每个方程只含一个未知量，即在上三角形方程组中，只保留对角线上的未知量$x_1, x_2, \cdots, x_n$。它们的系数均为1，则无需回代即可得到所求的解。这就是Jordan(约当)消去法。此法计算量大，乘除法计算次数为$n^3/2$，而高斯消去法只有$n^3/3$。因此，Jordan消去法用得较少，但用它来求逆矩阵却是很方便的。

6.2.2　列主元消去法

1. Guass消元法的条件

在Guass消元过程中，始终假设$a_{kk}^{(k-1)}\neq 0$，该元素$a_{kk}^{(k-1)}$被称为主元素。如果$a_{kk}^{(k-1)}=0$，显而易见，Guass消元过程无法继续进行下去。但是，如果$a_{kk}^{(k-1)}\neq 0$，而$|a_{kk}^{(k-1)}|$非常小，结果会导致舍入误差扩大、计算结果不可靠，从而严重降低精度，称这样的主元素$a_{kk}^{(k-1)}$为小主元素。因此，在Guass消元过程中，主元素不能为零或不能出现小主元素的情况。

例6.2　用Guass消去法解方程组：

$$\begin{cases}10^{-5}x_1+x_2=1\\ x_1+x_2=2\end{cases}\tag{A}$$

解　用Guass消元法求解该方程组的过程如下：对第1个方程，两边同时除$a_{11}^{(0)}=10^{-5}$得：

$$x_1+10^5x_2=10^5$$

用上述方程消去方程组的第2个方程的变量x_1，得：

$$(1-10^5)x_2=2-10^5$$

取4位浮点十进制进行计算，有：

$$1-10^5\approx-10^5,\quad 2-10^5\approx-10^5\text{(对阶过程)}$$

从而得到消元后的方程组：

$$\begin{cases}10^{-5}x_1+x_2=1\\ x_2=1\end{cases}\tag{B}$$

将$x_2=1$回代到方程组(B)的第1个方程$x_1+10^5x_2=10^5$，得：

$$x_1=0$$

此方程组的正确解应为$x_1=x_2=1$，而用Guass消元法得到的解$x_1=0, x_2=1$严重失真，其原因就在于$a_{11}=10^{-5}$太小了。如果将两个方程的位置调换一下，则用Guass消元法就可以得到正确解。即：

$$\begin{cases}x_1+x_2=2\\ 10^{-5}x_1+x_2=1\end{cases}\tag{C}$$

将方程组(C)第2个方程的变量x_1消去，得：

$$(1-10^{-5})x_2=1-2\times10^{-5}$$

$$x_2=1$$

消元后的方程组为：

$$\begin{cases}x_1+x_2=2\\ x_2=1\end{cases}\tag{D}$$

将 $x_2=1$ 回代到方程组(D)第 1 个方程 $x_1+x_2=2$ 中，得：

$$x_1=1$$

该结果是正确的。

在本例中，方程组(A)和方程组(C)虽然是等价的，但它们的主元素却是完全不同的。

定义 6.1　对于 $m\times n$ 阶矩阵 A，由 A 的既在前 k 行 $(1\leqslant k\leqslant \min(m,n))$ 又在前 k 列的 k^2 个元素，保持在矩阵 A 中的相对位置而组成的矩阵，称为矩阵 A 的 k 阶主子矩阵。k 阶主子矩阵对应的行列式

$$\Delta_k=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{vmatrix} \quad k=1,2,\cdots,n$$

称为矩阵 A 的 k 阶主子式或顺序主子式。

对于 n 阶线性方程组 $AX=B$，只要 $\det A\neq 0$，就可以用 Cramer 法则求出方程组的惟一解。然而，用 Guass 消去法求解线性方程组时，条件要严格得多。

从上面的讨论中可以得出结论：Guass 消去法能够进行到底的条件是矩阵 A 的所有顺序主子式 Δ_k 都不能为零。因此，当应用顺序消去法不能正确求出方程组的解时，应该使用改进的 Guass 消去法，即选主元的 Guass 消去法。本书只介绍按列选主元素的消去法。

2. 列主元消去法

对于线性方程组(6.4)，假设经过 $k-1$ 次消元后，得到 $k-1$ 个方程和 $n-(k-1)$ 个未经消元的方程：

$$\begin{cases} a_{11}^{(0)}x_1+a_{12}^{(0)}x_2+a_{13}^{(0)}x_3+\cdots+a_{1j}^{(0)}x_j+\cdots+a_{1n}^{(0)}x_n=b_1^{(0)} \\ \qquad a_{22}^{(1)}x_2+a_{23}^{(1)}x_3+\cdots+a_{2j}^{(1)}x_j+\cdots+a_{2n}^{(1)}x_n=b_2^{(1)} \\ \qquad\qquad \vdots \\ \qquad\qquad a_{kk}^{(k-1)}x_k+a_{k(k+1)}^{(k-1)}x_{k+1}+\cdots+a_{kn}^{(k-1)}x_n=b_k^{(k-1)} \\ \qquad\qquad \vdots \\ \qquad\qquad a_{lk}^{(k-1)}x_k+a_{l(k+1)}^{(k-1)}x_{k+1}+\cdots+a_{ln}^{(k-1)}x_n=b_l^{(k-1)} \\ \qquad\qquad \vdots \\ \qquad\qquad a_{nk}^{(k-1)}x_k+a_{n(k+1)}^{(k-1)}x_{k+1}+\cdots+a_{nn}^{(k-1)}x_n=b_n^{(k-1)} \end{cases}$$

如果

$$|a_{lk}^{(k-1)}|=\max_{k\leqslant i\leqslant n}|a_{ik}^{(k-1)}|$$

称 $a_{lk}^{(k-1)}$ 为列主元素，这时将方程组的第 l 行和第 k 行进行交换，然后再继续进行第 k 次消元过程。按照这种方法，一般能够将方程组的消元过程进行到底。这种消元方法被称为列主元消去法。

例 6.3　用高斯列主元消去法解下列方程组：

$$\begin{cases} 2x_1-x_2-3x_2=1 \\ 4x_1+2x_2+5x_3=4 \\ x_1+2x_2=7 \end{cases}$$

解　系数矩阵为：

$$A=\begin{bmatrix}a_{11} & a_{12} & a_{13}\\ a_{21} & a_{22} & a_{23}\\ a_{31} & a_{32} & a_{33}\end{bmatrix}=\begin{bmatrix}2 & -1 & 3\\ 4 & 2 & 5\\ 1 & 2 & 0\end{bmatrix}$$

对系数矩阵 A 选主元并进行消元，如表6.1所示。消元的结果得到三角形方程：

$$\begin{cases}x_1+0.5x_2+1.25x_3=1\\ \qquad x_2-0.25x_3=0.5\\ \qquad\qquad x_3=-6\end{cases}$$

回代后得解为：$x_3=-6$，$x_2=-1$，$x_1=9$

表6.1　选主元的消元过程

消元过程	系数矩阵 A	B	选主元素过程
	2　−1　3 4　2　5 1　2　0	1 4 7	在 a_{11}，a_{21}，a_{31} 中选出主元素 $a_{21}=4$
	4　2　5 2　−1　3 1　2　0	4 1 7	第1行和第2行互易位置
第1步消元，分离出 x_1，并从2，3方程中消去 x_1	1　0.5　1.25 0　−2　0.5 0　1.5　−1.25	1 −1 6	在 a_{22}，a_{32} 中选出主元素 $a_{22}=-2$，因 x_2 的系数就是主元素，故不必互易位置
第2步消元分离出 x_2，并从方程3中消去 x_2	1　0.5　1.25 0　1　−0.25 0　0　−0.875	1 0.5 6.25	主元素即为 $a_{33}=-0.875$
第3步消元分离出 x_3	1　0.5　1.25 0　1　−0.25 0　0　1	1 0.5 −6	

例6.4　在MATLAB计算环境中，用高斯消去法求解方程组：

$$\begin{bmatrix}2 & -1 & 3\\ 4 & 2 & 5\\ 1 & 2 & 0\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ x_3\end{bmatrix}=\begin{bmatrix}1\\ 4\\ 7\end{bmatrix}$$

解　首先定义一个增广矩阵：

```
a=[2  -1  3  1;
   4   2  5  4;
   1   2  0  7]
```

其中，前3列是系数矩阵，最后一列是方程式的右端常数项（增广的列）。第1次选主元，比较第1列的各元素之后可知，应将第1行与第2行交换，即：

```
tempo=a(2,:); a(2,:)=a(1,:); a(1,:)=tempo;
```

得

```
a=[4   2  5  4;
   2  -1  3  1;
```

```
    1    2  0  7]
```

下面对矩阵 a 进行消元：

```
a(2,:)=a(2,:)  a(1,:) * a(2,1)/a(1,1);
a(3,:)=a(3,:) - a(1,:) * a(3,1)/a(1,1);
```

得

```
a=
    4.0000    2.0000    5.0000    4.0000
         0   -2.0000    0.5000   -1.0000
         0    1.5000   -1.2500    6.0000
```

第 2 次选主元：比较第 2 列的各元素之后可知，列主元素就是 −2.0000，不必进行交换，继续对矩阵 a 进行消元：

```
a(3,:)=a(3,:) - a(2,:) * a(3,2)/a(2,2);
```

得

```
a=
    4.0000    2.0000    5.0000    4.0000
    0        -2.0000    0.5000   -1.0000
    0         0        -0.8750    5.2500
```

消去过程完成。回代过程如下：

```
x(3)=a(3,4)/a(3,3);
x(2)=(a(2,4) - a(2,3) * x(3))/a(2,2);
x(1)=(a(1,4) - a(1,2) * x(2) - a(1,3) * x(3))/a(1,1);
```

最后，方程组的解为：

```
x=
     9
    -1
    -6
```

6.2.3　矩阵三角分解法

方程组(6.4)通过 Guass 消去法变成方程组(6.8)，即从 $AX=B$ 的形式等价变成 $UX=Y$ 的形式。从矩阵运算的意义上讲，这是对矩阵 A 实施某种 P 变换而得到矩阵 U。或者说，矩阵 A 可以分解成矩阵 L 和矩阵 U 的乘积。那么，矩阵 P 和 L 到底是什么呢？下面就来讨论这个问题。

1. 矩阵三角分解的原理

设方程组 $AX=B$ 的系数矩阵 A 的顺序主子式不为零，即：

$$\Delta_k=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \vdots & \vdots & & \vdots \\ a_{k1} & a_{k2} & \cdots & a_{kk} \end{vmatrix} \neq 0, \quad k=1,2,\cdots,n$$

在 Guass 消元法中，第 1 次消元时，相当于用单位下三角阵：

$$L_1^{-1}=\begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ -m_{21} & 1 & 0 & \cdots & 0\\ -m_{31} & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ -m_{n1} & 0 & 0 & \cdots & 1\end{bmatrix}$$

左乘方程组 $AX=B$，从而得到：

$$A_1X=B_1$$

其中

$$A_1=L_1^{-1}A=\begin{bmatrix}a_{11} & a_{12} & \cdots & a_{1n}\\ 0 & a_{22}^{(1)} & \cdots & a_{2n}^{(1)}\\ \vdots & \vdots & & \vdots\\ 0 & a_{n2}^{(1)} & \cdots & a_{nn}^{(1)}\end{bmatrix}$$

$$B=L_1^{-1}B=(a_{1,n+1},a_{2,n+1}^{(1)},\cdots,a_{n,n+1}^{(1)})^T$$

第2次消元时，相当于用单位下三角阵：

$$L_2^{-1}=\begin{bmatrix}1 & 0 & 0 & 0 & \cdots & 0\\ 0 & 1 & 0 & 0 & \cdots & 0\\ 0 & -m_{32} & 1 & 0 & \cdots & 0\\ 0 & -m_{42} & 0 & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & -m_{n2} & 0 & 0 & \cdots & 1\end{bmatrix}$$

左乘方程组 $A_1X=B_1$，从而得到：

$$A_2X=B_2$$

其中

$$A_2=L_2^{-1}L_1^{-1}A=\begin{bmatrix}a_{11} & a_{12} & a_{13} & \cdots & a_{1n}\\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)}\\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{2n}^{(2)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)}\end{bmatrix}$$

$$B_2=L_2^{-1}L_1^{-1}B=(a_{1,n+1},a_{2,n+1}^{(1)},a_{3,n+1}^{(2)},\cdots,a_{n,n+1}^{(2)})^T$$

重复上述过程，经过 $n-1$ 次消元后，最后得到等价方程组：

$$A_{n-1}X=B_{n-1}$$

其中

$$A_{n-1}=L_{n-1}^{-1}L_{n-2}^{-1}\cdots L_2^{-1}L_1^{-1}A=\begin{bmatrix}a_{11} & a_{12} & a_{13} & \cdots & a_{1n}\\ 0 & a_{22}^{(1)} & a_{23}^{(1)} & \cdots & a_{2n}^{(1)}\\ 0 & 0 & a_{33}^{(2)} & \cdots & a_{3n}^{(2)}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & 0 & 0 & a_{nn}^{(n-1)}\end{bmatrix}$$

$$B_{n-1}=L_{n-1}^{-1}L_{N-1}^{-1}\cdots L_2^{-1}L_1^{-1}B=(a_{1,n-1},a_{2,n-1}^{(1)},\cdots,a_{n,n-1}^{(n-1)})^T$$

可见，A_{n-1} 实际上就是一个上三角矩阵，可以记为：

$$U=A_{n-1}=L_{n-1}^{-1}L_{n-2}^{-1}\cdots L_2^{-1}L_1^{-1}A$$

所以

$$A=(L_1L_2\cdots L_{n-1})U=LU \tag{6.10}$$

其中,$L=L_1L_2\cdots L_{n-1}$。不难验证:

$$L=\begin{bmatrix}1&0&0&\cdots&0\\m_{21}&1&0&\cdots&0\\m_{31}&m_{32}&1&\cdots&0\\\vdots&\vdots&\vdots&\ddots&\vdots\\m_{n1}&m_{n2}&\cdots&m_{nn-1}&1\end{bmatrix}$$

显然,L 为单位下三角矩阵。称$A=LU$ 为矩阵A 的LU 分解或三角分解。于是解线性方程组$AX=B$,就转化为解方程组 $LUX=B$。若令$UX=Y$,就得到一个与 $AX=B$ 等价的方程组:

$$\begin{bmatrix}1&0&0&\cdots&0\\l_{21}&1&0&\cdots&0\\l_{31}&l_{32}&1&\cdots&0\\\vdots&\vdots&\vdots&&\vdots\\l_{n1}&l_{n2}&l_{n3}&\cdots&1\end{bmatrix}\begin{bmatrix}y_1\\y_2\\y_3\\\cdots\\y_n\end{bmatrix}=\begin{bmatrix}b_1\\b_2\\b_3\\\cdots\\b_n\end{bmatrix}$$

和

$$\begin{bmatrix}u_{11}&u_{12}&u_{13}&\cdots&u_{1n}\\0&u_{22}&u_{23}&\cdots&u_{2n}\\0&0&u_{33}&\cdots&u_{3n}\\\vdots&\vdots&\vdots&&\vdots\\0&0&0&\cdots&u_{nn}\end{bmatrix}=\begin{bmatrix}x_1\\x_2\\x_3\\\cdots\\x_n\end{bmatrix}=\begin{bmatrix}y_1\\y_2\\y_3\\\cdots\\y_n\end{bmatrix}$$

这两个矩阵方程很容易利用回代法求解:先解出Y,即:

$$y_i=b_i-\sum_{j=1}^{i-1}l_{ij}y_i,\qquad i=1,2,\cdots,n$$

然后就可以解出X:

$$x_i=\left(y_i-\sum_{j=i+1}^{n}u_{ij}x_j\right)\Big/u_{ii},\qquad i=n,n-1,\cdots,2,1$$

在MATLAB 语言中,方程组$AX=B$ 的求解过程十分简洁。即

$$Y=L\backslash B$$
$$X=Y\backslash U$$

上述解方程组的方法称为解线性方程组的三角直接分解法。这种方法实际上是Guass 消去法的一种变形。矩阵L 和U 的各元素可根据矩阵乘法、比较 $A=LU$ 两边的系数求出。

关于矩阵的三角分解,有下面的定理:

定理6.1　若A 为n 阶方阵,且A 的所有顺序主子式$\Delta_k\neq 0$,$(k=1,2,\cdots,n)$,则存在惟一的一个单位下三角矩阵L 和一个上三角矩阵U,使得$A=LU$ 成立。

证明　在矩阵的三角分解原理中,已经推导了矩阵的三角分解过程,此处不再复述。下面只需证明矩阵的这种分解过程是惟一的。

用反证法证明分解的惟一性。若另有分解

$$A=\tilde{L}\tilde{U}$$

则

$$A=LU=\tilde{L}\tilde{U}$$

由于 A 是非奇异矩阵，所以 $\tilde{L}$ 及 $\tilde{U}$ 也是非奇异矩阵，因而

$$U\tilde{U}^{-1}=L^{-1}\tilde{L}$$

而 $L^{-1}\tilde{L}$ 是下三角矩阵且主对角线元素全为1，$U\tilde{U}^{-1}$ 为上三角矩阵，故上式成立的充要条件是：

$$U\tilde{U}^{-1}=L^{-1}\tilde{L}=E \quad (E\text{ 是 } n \text{ 阶单位阵})$$

从而 $U=\tilde{U}$，$L=\tilde{L}$。惟一性得到证明。

若矩阵 A 非奇异，但不满足所有的顺序主子式都不为零这一条件，那么应用Guass 列主元消去法，也可以将矩阵 A 分解。即：

第1次消元：将第1行与列主元素 $a_{r_1 1}^{(0)}$ 所在的第 r_1 行进行交换，相当于将方程组 $AX=B$ 左乘矩阵 P_{1r_1}，即：

$$P_{1r_1}AX=P_{1r_1}B$$

经第1次消元后，得：

$$L_1^{-1}P_{1r_1}AX=L_1^{-1}P_{1r_1}B$$

即系数矩阵为：

$$A_1=L_1^{-1}P_{1r_1}A$$

其中

$$P_{1r_1}=\begin{bmatrix} 0 & 0 & \cdots & 0 & 0 & 1 & \cdots & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & \ddots & \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & 1 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & \cdots & \cdots & 0 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & \cdots & \cdots & 0 & 0 & 0 & \cdots & \cdots & 0 \\ \vdots & \cdots & \cdots & \vdots & \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & \cdots & \cdots & 0 & 0 & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{matrix} \text{第1行} \\ \\ \\ \\ \\ \text{第}r_1\text{行} \\ \\ \\ \\ \end{matrix}$$

类似地，经 $n-1$ 次消元后，有：

$$A_{n-1}=L_{n-1}^{-1}P_{n-1,r_{n-1}}\cdot L_{n-2}^{-1}P_{n-2,r_{n-2}}\cdots L_1^{-1}P_{1,r_1}A$$

如果预先知道每一个 P_{ir_i}，$(i=1,2,\cdots,n-1)$，则在消元之前就全部作交换，即：

$$P_{n-1,r_{n-1}}\cdot P_{n-2,r_{n-2}}\cdots P_{1,r_1}A=PA$$

其中，$P=P_{n-1,r_{n-1}}\cdot P_{n-2,r_{n-2}}\cdots P_{1,r_1}$，即原方程变为：

$$PAX=PB$$

然后再消元，相当于对 PA 做三角分解 $PA=LU$。称矩阵 A 的这种分解为 PLU 分解，并有以下定理：

定理6.2　若 A 非奇异，则一定存在置换矩阵 P，使得 PA 被分解为一个单位下三角阵 L 和一个上三角阵 U 的乘积，即 $PA=LU$ 成立。

这时，原方程组 $AX=B$ 等价于：

$$PAX=PB$$

即等价于求解

$$LUX=PB$$

令
$$UX=Y$$
则
$$LY=PB,\quad Y=L^{-1}PB$$
$$UX=Y,\quad X=U^{-1}Y$$

2. Doolittle(杜里特尔)分解法

假设系数矩阵 A 不需要进行交换且三角分解是惟一的,则 $A=LU$ 可以分解如下:

$$L=\begin{bmatrix}1 & 0 & \cdots & 0\\ l_{21} & 1 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ l_{n1} & l_{n2} & \cdots & 1\end{bmatrix},\quad U=\begin{bmatrix}u_{11} & u_{12} & \cdots & u_{1n}\\ 0 & u_{22} & \cdots & u_{2n}\\ \vdots & \vdots & \ddots & \vdots\\ 0 & \cdots & 0 & u_{nn}\end{bmatrix}$$

于是有:

$$\begin{bmatrix}a_{11} & a_{12} & a_{13} & \cdots & a_{1n}\\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n}\\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n}\\ \vdots & \vdots & \vdots & & \vdots\\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn}\end{bmatrix}=\begin{bmatrix}1 & 0 & 0 & \cdots & 0\\ l_{21} & 1 & 0 & \cdots & 0\\ l_{31} & l_{32} & 1 & \cdots & 0\\ \vdots & \vdots & \vdots & & \vdots\\ l_{n1} & l_{n2} & l_{n3} & \cdots & 1\end{bmatrix}\begin{bmatrix}u_{11} & u_{12} & u_{13} & \cdots & u_{1n}\\ 0 & u_{22} & u_{23} & \cdots & u_{2n}\\ 0 & 0 & u_{33} & \cdots & u_{3n}\\ \vdots & \vdots & \vdots & & \vdots\\ 0 & 0 & 0 & \cdots & u_{nn}\end{bmatrix}\tag{6.11}$$

矩阵的这种 LU 分解称为Doolittle 分解。

为了计算L 和U 矩阵,由于系数矩阵A 是已知的,所以可以通过比较上式的两边,逐步把L 和U 构造出来。

设a_{ij}为矩阵 A 的第i 行第j 列元素,根据矩阵乘法,有:

$$a_{ij}=\sum_{p=1}^{n}l_{ip}u_{pj}\quad (i=1,2,\cdots,n;j=1,2,\cdots,n)$$

根据L 和U 矩阵的特点,可以将上式写成:

当$i=j$ 时,$a_{ii}=\sum_{p=1}^{i-1}l_{ip}u_{pj}+u_{ii}\quad (i=j=1,2,\cdots,n)$

当$i<j$ 时,$a_{ij}=\sum_{p=1}^{i-1}l_{ip}u_{pj}+u_{ij}\quad (j=2,3,\cdots,n;i=1,2,\cdots,j-1)$

当$i>j$ 时,$a_{ij}=\sum_{p=1}^{j}l_{ip}u_{pj}\quad (i=2,3,\cdots,n;j=1,2,\cdots,i-1)$

或者　$a_{ij}=\sum_{p=1}^{j-1}l_{ip}u_{pj}+l_{ij}u_{jj}\quad (i=2,3,\cdots,n;j=1,2,\cdots,i-1)$

根据上述公式,按照表6.2 所示的计算顺序,逐层计算L 和U 矩阵的各元素。

表6.2　LU 矩阵元素的计算顺序表

行＼列	第1列	第2列	第3列	…	第n列	
第1行	u_{11}	u_{12}	u_{13}	…	u_{1n}	第1层
第2行	l_{21}	u_{22}	u_{23}	…	u_{2n}	第2层

续表

行＼列	第1列	第2列	第3列	…	第 n 列	
第3行	l_{31}	l_{32}	u_{33}	…	u_{3n}	第3层
⋮	⋮	⋮	⋮	…	⋮	⋮
第 n 行	l_{n1}	l_{n2}	l_{n3}	…	u_{nn}	第 n 层
	第1层	第2层	第3层	…	第 n 层	计算层次

第一步：计算第1层的元素，即求出 u_{11} 和第1行的 u_{1j}，第1列的 l_{i1}。

由 L 阵的第1行乘 U 阵的第1列，先算出 U 阵第1行的元素 u_{11}，由 L 阵的第1行分别乘 U 阵的各列，算出 U 阵第1行的元素 u_{1j}，再由 L 阵的各行分别去乘 U 阵的第1列，算出 L 阵第1列的元素 l_{i1}：

$$\begin{cases} u_{11}=a_{11} & (j=1) \\ u_{1j}=a_{1j} & (j=2,\cdots,n) \\ l_{i1}=a_{i1}/u_{11} & (i=2,3,\cdots,n) \end{cases} \tag{6.12}$$

第二步：计算第 k 层的元素。现假设已经算出 U 阵的前 $k-1$ 行元素，L 阵的前 $k-1$ 列元素，下面来计算 U 阵的第 k 行元素，L 阵的第 k 列元素，即计算 u_{kk}、u_{kj} 和 l_{ik}。

由 L 阵的第 k 行乘 U 阵的第 k 列得 U 阵第 k 行的元素 u_{kk}，由 L 阵的第 k 行分别乘 U 阵的第 j 列（$j=k+1,\cdots,u$）得 U 阵第 k 行的元素 u_{kj}，再由 L 阵的第 i 行（$i=k+1,k+2,\cdots,n$）分别去乘 U 阵的第 k 列得 L 阵第 k 列的元素 l_{ik}：

$$\begin{cases} u_{kk}=a_{kk}-\sum\limits_{p=1}^{k-1}l_{kp}u_{pk} & (k=2,3,\cdots,n) \\ u_{kj}=a_{kj}-\sum\limits_{p=1}^{k-1}l_{kp}u_{pj} & (k=2,3,\cdots,n-1;j=k+1,\cdots,n) \\ l_{ik}=\dfrac{\left(a_{ik}-\sum\limits_{p=1}^{k-1}l_{ip}u_{pk}\right)}{u_{kk}} & (k=2,3,\cdots,n-1;j=k+1,k+2,\cdots,n) \end{cases} \tag{6.13}$$

根据方程组(6.12)和(6.13)，逐层计算，就可完成矩阵 L 和 U 各元素的计算。

例6.5 用矩阵分解法解方程组：

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 2 \\ 3 & 1 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}=\begin{pmatrix} 6 \\ 9 \\ 9 \end{pmatrix}$$

解 矩阵 A 按Doolittle分解法进行分解：

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 2 \\ 3 & 1 & 5 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{pmatrix}\begin{pmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{pmatrix}$$

矩阵U和L的计算过程如下：

$$u_{11}=a_{11}=1$$
$$l_{21}u_{11}=a_{21}=2 \qquad l_{21}=2$$
$$l_{31}u_{11}=a_{31}=3 \qquad l_{31}=3$$
$$u_{12}=a_{12}=2$$
$$u_{13}=u_{13}=3$$
$$u_{12}l_{21}+u_{22}=a_{22}=5 \qquad u_{22}=1$$
$$l_{31}u_{12}+l_{32}u_{22}=a_{32}=1 \qquad l_{32}=-5$$
$$l_{21}u_{13}+u_{23}=a_{23}=2 \qquad u_{23}=-4$$
$$l_{31}u_{13}+l_{32}u_{23}+u_{33}=a_{33}=5 \qquad u_{33}=-24$$

分解结果为：

$$\begin{pmatrix}1 & 2 & 3\\ 2 & 5 & 2\\ 3 & 1 & 5\end{pmatrix}=\begin{pmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & -5 & 1\end{pmatrix}\begin{pmatrix}1 & 2 & 3\\ 0 & 1 & -4\\ 0 & 0 & -24\end{pmatrix}$$

先解方程$LY=B$,即：

$$\begin{pmatrix}1 & 0 & 0\\ 2 & 1 & 0\\ 3 & -5 & 1\end{pmatrix}\begin{pmatrix}y_1\\ y_2\\ y_3\end{pmatrix}=\begin{pmatrix}6\\ 9\\ 9\end{pmatrix} \quad 得 \quad \begin{pmatrix}y_1\\ y_2\\ y_3\end{pmatrix}=\begin{pmatrix}6\\ -3\\ -24\end{pmatrix}$$

再解方程$UX=Y$,即：

$$\begin{pmatrix}1 & 2 & 3\\ 0 & 1 & -4\\ 0 & 0 & -24\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}=\begin{pmatrix}6\\ -3\\ -24\end{pmatrix} \quad 得 \quad \begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix}=\begin{pmatrix}1\\ 1\\ 1\end{pmatrix}$$

例 6.6　用LU分解法求解线性方程组$AX=B$,其中：

$$A=\begin{bmatrix}2 & 1 & -3\\ -1 & 3 & 2\\ 3 & 1 & -3\end{bmatrix} \quad B=\begin{bmatrix}2\\ 0\\ 1\end{bmatrix}$$

解一　将A分解为：

$$L=\begin{bmatrix}1 & 0 & 0\\ -0.5 & 1 & 0\\ 1.5 & -0.1428 & 1\end{bmatrix} \quad U=\begin{bmatrix}2 & 1 & -3\\ 0 & 3.5 & 0.5\\ 0 & 0 & 1.5714\end{bmatrix}$$

首先求解$LY=B$,即：

$$\begin{bmatrix}1 & 0 & 0\\ 0.5 & 1 & 0\\ 1.5 & -0.1428 & 1\end{bmatrix}\begin{bmatrix}y_1\\ y_2\\ y_3\end{bmatrix}=\begin{bmatrix}2\\ 0\\ 1\end{bmatrix}$$

解之得：

$$y_1=2$$
$$y_2=0+2\times(0.5)=1$$
$$y_3=1-2\times1.5-0.1\times(-0.1428)=-1.8572$$

然后求$UX=Y$,即：

$$\begin{bmatrix} 2 & 1 & -3 \\ 0 & 3.5 & 0.5 \\ 0 & 0 & 1.5714 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ -1.8572 \end{bmatrix}$$

得到原方程组的解为：

$$x_3 = -1.8572/1.5714 = -1.1818$$

$$x_2 = (1-0.5x_3)/3.5 = 0.4545$$

$$x_1 = (2-x_1-3x_3)/2 = -1$$

解二 在MATLAB中，A 的 LU 分解函数是lu，这个函数的调用格式通常是：

[L,U]=lu(A);

其中，A 是待分解的矩阵；L，U 分别代表前面讨论的 L 和 U。

```
A=[2,1,-3; -1,3,2; 3,1,-3]; B=[2; 0; 1]
[L,U]=lu (A);
```

其结果为：

```
L=
    1.0000    0         0
    -0.3333   1.0000    0
    0.6667    0.1000    1.0000
U=
    3.0000    1.0000    -3.0000
    0         3.3333    1.0000
    0         0         -1.1000
```

首先求解 Y，即：

```
Y=L\B;
```

得

```
Y=
1.0000
0.3333
1.3000
```

然后求解 X，即：

```
X=U\Y;
```

得

```
X=
-1.0000
0.4545
-1.1818
```

解三 在MATLAB中，方程组的解也可以直接由下列矩阵运算得到，即：

$AX=B$;　　$X=A\backslash B$

$AX=B$;　　$X=inv(A)^{*}B$

3. Crout(克洛特)分解法

假设系数矩阵 A 不需要进行交换，且三角分解是惟一的。Crout 分解法也是一种直接分解，即 $A=\hat{L}\hat{U}$，与Doolittle 分解法的区别在于，$\hat{L}$ 是一般下三角阵，而 $\hat{U}$ 是一个单位上三角阵。即：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} \hat{l}_{11} & 0 & \cdots & 0 \\ \hat{l}_{21} & \hat{l}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{l}_{n1} & \hat{l}_{n2} & \cdots & \hat{l}_{nn} \end{bmatrix} \begin{bmatrix} 1 & \hat{u}_{12} & \cdots & \hat{u}_{1n} \\ 0 & 1 & \cdots & \hat{u}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.14}$$

比较式(6.14)的两边,就可以推出矩阵$\hat{L}$和$\hat{U}$的计算公式。Crout 分解方法是,先计算$\hat{L}$的第r列,然后再计算$\hat{U}$的第r行。

第一步:先计算$\hat{L}$矩阵的第1列。由$\hat{L}$阵的各行分别去乘$\hat{U}$阵的第1列,得:

$$\hat{l}_{i1}=a_{i1} \qquad i=1,2,\cdots,n \tag{6.15}$$

第二步:再计算$\hat{U}$矩阵的第1行。由$\hat{L}$阵的第1行分别去乘$\hat{U}$阵的各列,得:

$$\hat{u}_{1j}=a_{1j}/\hat{l}_{11} \qquad j=2,3,\cdots,n \tag{6.16}$$

若已计算出由$\hat{L}$矩阵的前$r-1$列元素和$\hat{U}$矩阵的前$r-1$行元素,则可得到$\hat{L}$矩阵的第r列元素和$\hat{U}$矩阵的第r行元素,即:

$$\begin{cases} \hat{l}_{ir}=a_{ir}-\sum\limits_{k=1}^{r-1}\hat{l}_{ik}\hat{u}_{kr} & i=r,r+1,\cdots,n \\ \hat{u}_{rj}=\left(a_{rj}-\sum\limits_{k=1}^{r-1}\hat{l}_{rk}\hat{u}_{kj}\right) & j=r+1,r+2,\cdots,n \end{cases} \tag{6.17}$$

这里$r=1,2,3,\cdots,r$。于是,方程组$AX=B$等价于:

$$\begin{cases} \hat{L}Y=B \\ \hat{U}X=Y \end{cases}$$

通过逐次向前代入过程,得$\hat{L}Y=B$的解为:

$$\begin{cases} y_1=b_1/\hat{l}_{11} \\ y_i=\left(b_i-\sum\limits_{j=1}^{i-1}\hat{l}_{ij}y_j\right)/\hat{l}_{ii} & i=2,3,\cdots,n \end{cases} \tag{6.18}$$

再通过逐次向后回代过程,得$\hat{U}X=Y$的解为:

$$\begin{cases} x_n=y_n \\ x_i=y_i-\sum\limits_{j=i+1}^{n}\hat{u}_{ij}x_j & i=n-1,\cdots,2,1 \end{cases} \tag{6.19}$$

4. Cholesky(乔列斯基)分解法

若矩阵A的三角分解是惟一的,则通过比较矩阵A的 Doolittle 分解矩阵U和 Crout 分解矩阵$\hat{L}$的元素可知,两者对角线上的元素是相等的,即:

$$u_{rr}=\hat{l}_{rr} \qquad r=1,2,\cdots,n$$

若令

$$D=\mathrm{diag}(u_{11},u_{22},\cdots,u_{nn})$$

则有

$$A=LU=(LD)(D^{-1}U)=\hat{L}\hat{U}$$

因此,矩阵A的分解可写成:

$$A=LD\hat{U} \tag{6.20}$$

其中,L是单位下三角阵,$\hat{U}$是单位上三角阵,D是对角阵。矩阵A的这种分解也是惟一的。

若A为对称正定矩阵,则在(6.20)中有$\hat{U}=L^T$,于是有:

$$A=LD\hat{U}=LDL^T=(LD^{\frac{1}{2}})(LD^{\frac{1}{2}})^T=\bar{L}\bar{L}^T \tag{6.21}$$

其中，$\bar{L}$ 为下三角阵，将(6.21)式展开，有：

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}=\begin{bmatrix} \bar{l}_{11} & 0 & \cdots & 0 \\ \bar{l}_{21} & \bar{l}_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \bar{l}_{n1} & \bar{l}_{n2} & \cdots & \bar{l}_{nn} \end{bmatrix}\begin{bmatrix} \bar{l}_{11} & \bar{l}_{21} & \cdots & \bar{l}_{n1} \\ 0 & \bar{l}_{22} & \cdots & \bar{l}_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \bar{l}_{nn} \end{bmatrix} \tag{6.22}$$

根据矩阵乘法，比较(6.22)式两边，有：

$$a_{rr}=\sum_{k=1}^{r-1}\bar{l}_{rk}^2+\bar{l}_{rr}^2 \qquad r=1,2,\cdots,n$$

$$a_{ir}=\sum_{k=1}^{r-1}\bar{l}_{ik}\bar{l}_{rk}+\bar{l}_{ir}\bar{l}_{rr} \qquad r=1,2,\cdots,i-1$$

于是，得到矩阵 $\bar{L}$ 元素的计算公式为：

$$\begin{aligned} \bar{l}_{rr}&=\left(a_{rr}-\sum_{k=1}^{r-1}\bar{l}_{rk}^2\right)^{\frac{1}{2}} \\ \bar{l}_{ir}&=\left(a_{ir}-\sum_{k=1}^{r-1}\bar{l}_{ik}\bar{l}_{rk}\right)\Big/\bar{l}_{rr} \end{aligned} \tag{6.23}$$

其中，$r=1,2,\cdots,n$；$i=r+1,r+2,\cdots,n$。按照 $\bar{l}_{11}\to\bar{l}_{21}\to\bar{l}_{22}\to\bar{l}_{31}\to\bar{l}_{32}\to\bar{l}_{33}\to\cdots$ 的计算顺序，矩阵 $\bar{L}$ 的元素完全可以计算出来。称矩阵 $A=\bar{L}\bar{L}^T$ 的这种分解为Cholesky 分解。于是，方程组 $AX=B$ 可分解成 $\bar{L}Y=B$ 和 $\bar{L}^TX=Y$ 两个方程组来求解，即：

$$\begin{cases} y_1=b_1/\bar{l}_{11} \\ y_i=\left(b_i-\sum_{k=1}^{i-1}\bar{l}_{ik}y_k\right)\Big/\bar{l}_{ii} \qquad i=2,3,\cdots,n \end{cases} \tag{6.24}$$

和

$$\begin{cases} x_n=y_n/\bar{l}_{nn} \\ x_i=\left(y_i-\sum_{k=i+1}^{n}\bar{l}_{ki}x_k\right)\Big/\bar{l}_{ii} \qquad i=n-1,n-2,\cdots,2,1 \end{cases} \tag{6.25}$$

上述解方程组的方法，称为Cholesky 分解方法或平方根法。

Cholesky 分解的缺点是需要进行开平方运算。为避免开平方运算可以作如下分解：

$$A=LDL^T \tag{6.26}$$

即

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}=\begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}\begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}\begin{bmatrix} 1 & l_{21} & \cdots & l_{n1} \\ & 1 & \cdots & l_{n2} \\ & & \ddots & \vdots \\ & & & 1 \end{bmatrix} \tag{6.27}$$

根据矩阵乘法，比较(6.27)式两边，即：

$$a_{ir}=\sum_{k=1}^{r-1}l_{ik}d_kl_{rk}+l_{ir}d_r \qquad i=2,3,\cdots,n;r=1,2,\cdots,i$$

于是，得到矩阵 D 和 L 元素的计算公式为：

$$\begin{cases} d_r=a_{rr}-\sum_{k=1}^{r-1}l_{rk}^2d_k \\ l_{ir}=\left(a_{ir}-\sum_{k=1}^{r-1}l_{ik}d_kl_{rk}\right)\Big/d_r \end{cases} \tag{6.28}$$

其中，$r=1,2,\cdots,n$；$i=r+1,r+2,\cdots,n$。按照$d_1\to d_{21}\to d_2\to d_{31}\to d_{32}\to d_3\to\cdots$的计算顺序，矩阵 D 和 L 的元素完全可以计算出来。称矩阵 $A=LDL^T$ 的这种分解为改进的Cholesky 分解。于是，方程组 $AX=B$ 可分解成$LY=B$ 和 $L^TX=D^{-1}Y$ 两个方程组来求解，即：

$$\begin{cases} y_1=b_1 \\ y_i=b_i-\sum\limits_{k=1}^{i-1}l_{ik}y_k \qquad i=2,3,\cdots,n \end{cases} \tag{6.29}$$

和

$$\begin{cases} x_n=y_n/d_n \\ x_i=y_i/d_i-\sum\limits_{k=i+1}^{n}l_{ki}x_k \qquad i=n-1,n-2,\cdots,2,1 \end{cases} \tag{6.30}$$

上述解方程组的方法，称为改进的Cholesky 分解方法或改进的平方根法。

Cholesky 分解方法的优点是不用选主元。由

$$a_{rr}=\sum_{k=1}^{r}l_{rk}^2$$

可知

$$|l_{rk}|\leqslant\sqrt{a_{rr}} \qquad k=1,2,\cdots,n$$

中间量l_{rk}得以控制，不会产生由中间量放大而使计算出现不稳定的现象。

在求解大型方程组时，节省存储单元是很重要的。由于矩阵 A 是对称的，只要将其下三角阵逐行存入一维数组 $A(1:M)$中即可，其中：

$$M=1+2+3+\cdots+n=n(n+1)/2$$

设 S 表示矩阵 A 的下三角部分前 $i-1$ 行元素的和，则

$$S=i(i-1)/2$$

因此，a_{ij}在$A(1:M)$中显然是第$S+J$ 个元素，即$a_{ij}(i\leqslant j)$存放于$A(S+J)$单元中。设有一个 $n=3$ 的正定对称线性方程组，则系数矩阵 A 为：

$$A=\begin{bmatrix} a_{11} & 0 & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad M=\frac{3(3+1)}{2}=6$$

故一维数组为 $A(1:6)$，有 6 个元素。元素 a_{32}应存放在下列位置单元中：

$$A(S+J)=A\left(\frac{3(3-1)}{2}+2\right)=A(5)$$

一般地，在程序运行过程中，逐行算出l_{ij}和d_i 后，l_{ij}和d_i 对应于系数矩阵 A 的存放位置为：

$$\begin{bmatrix} a_{11} & & & \\ a_{21} & a_{22} & & \\ \vdots & \vdots & \ddots & \\ a_{i1} & a_{i2} & \cdots & a_{ij} \end{bmatrix} \Rightarrow \begin{bmatrix} d_1 & & & \\ l_{21} & d_2 & & \\ \vdots & \vdots & \ddots & \\ l_{i1} & l_{i2} & \cdots & d_i \end{bmatrix}$$

例如，$n=3$；$\begin{bmatrix} a_{11} & & \\ a_{21} & a_{22} & \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \Rightarrow \begin{bmatrix} d_1 & & \\ l_{21} & d_2 & \\ l_{32} & l_{32} & d_3 \end{bmatrix}$

例 6.7　用乔列斯基方法解下列线性方程组：

$$\begin{cases}4x_1-2x_2-4x_3=10\\-2x_1+17x_2+10x_3=3\\-4x_1+10x_2+9x_3=-7\end{cases}$$

解　该方程组的系数矩阵 A 为正定对称阵，即：

$$A=\begin{bmatrix}a_{11}&&\\a_{21}&a_{22}&\\a_{31}&a_{32}&a_{33}\end{bmatrix}=\begin{bmatrix}4&&\\-2&17&\\-4&10&9\end{bmatrix}$$

计算 l_{ij} 和 $d_i(i=1,2,3;j=1,\cdots,i-1)$：

$$\begin{aligned}&i=1, && d_1=a_{11}=4\\&i=2,j=1, && l_{21}=(a_{21}-0)/d_1=-2/4=-1/2\\& && d_2=(a_{22}-d_1l_{21}^2)=17-4(1/2)^2=16\\&i=3,j=1 && l_{31}=(a_{31}-0)/d_1=-4/4=-1\\&\quad\ \ j=2 && l_{32}=(a_{32}-d_1l_{31}l_{21})/d_2=(10-4(-1/2)^2(-1))/16=1/2\\& && d_3=a_{33}-d_1l_{31}^2-d_2l_{32}^2=9-4(-1)^2-16(1/2)^2=1\end{aligned}$$

计算 $y_i(i=1,2,3;k=1,\cdots,i-1)$：

$$y_1=b_1=10$$

$$y_2=b_2-l_{21}y_1=3-(-1/2)\cdot 10=8$$

$$y_3=b_3-l_{31}y_1-l_{32}y_2=-7-(-1)(10)(-1/2)(8)=-1$$

计算 $x_i(i=3,2,1;k=i+1,\cdots,3)$：

$$x_3=y_3/d_3-0=-1/1=-1$$

$$x_2=y_2/d_2-l_{32}x_3=8/16-(1/2)-(-1)=1$$

$$x_1=y_1/d_1-l_{21}x_2-l_{31}x_3=10/4-(-1/2)(1)-(-1)(-1)=5/2+1/2-1=2$$

6.2.4　解三对角方程组的追赶法

在实际数值计算问题中，例如三次样条函数插值、用差分方法解常微分方程的边值问题等，常常遇到下列形式的方程组：

$$\begin{bmatrix}b_1&c_1&&&&&\\a_1&b_2&c_2&&&&\\&\ddots&\ddots&\ddots&&&\\&&a_i&b_i&c_i&&\\&&&\ddots&\ddots&\ddots&\\&&&&a_{n-1}&b_{n-1}&c_{n-1}\\&&&&&a_n&b_n\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\vdots\\x_i\\\vdots\\x_{n-1}\\x_n\end{bmatrix}=\begin{bmatrix}d_1\\d_2\\\vdots\\d_i\\\vdots\\d_{n-1}\\d_n\end{bmatrix}\tag{6.31}$$

用矩阵表示为 $AX=D$。这类方程组的系数矩阵为三对角矩阵，是一种带状的稀疏矩阵，其非零元素集中分布在主对角线及其相邻两条次对角线上，并满足下列条件：

$$\begin{cases}|b_1|>|c_1|\\|b_i|>|a_i|+|c_i| & a_i,c_i\neq 0,i=2,3,\cdots,n-1\\|b_n|>|c_n|\end{cases}$$

称这种特殊形式的线性方程组为三对角方程组。可以用Guass 消元法来求其全部的解。

1. 消元过程

先将方程组(6.31)中第1个方程 x_1 的系数化为1,得:

$$x_1+u_1x_2=y_1$$

式中,$u_1=c_1/b_1$,$y_1=d_1/b_1$。在方程组(6.31)中,除了第1个方程外,只有第2个方程中含有变量 x_2,所以消元过程很简单,于是有:

$$x_2+u_2x_3=y_2$$

式中,$u_2=c_2/(b_2-u_1a_2)$,$y_2=(d_2-y_1a_2)/(b_2-u_1a_2)$。依次顺序做下去,直到第 $k-1$ 个方程,有:

$$x_{k-1}+u_{k-1}x_k=y_{k-1}$$

利用上式消去第 k 个方程中的 x_{k-1},即:

$$(b_k-u_{k-1}a_k)x_k+c_kx_{k+1}=d_k-y_{k-1}a_k$$

令
$$u_k=\frac{c_k}{b_k-u_{k-1}a_k},\quad y_k=\frac{d_k-y_{k-1}a_k}{b_k-u_{k-1}a_k}$$

得
$$x_k+u_kx_{k+1}=y_k$$

因此,一直做下去,当 $k=n-1$ 时,便有:

$$x_{n-1}+u_{n-1}x_n=y_{n-1}$$

用上式消去方程组(6.31)最后一个方程的变量 x_{n-1},得:

$$x_n=y_n$$

式中 $y_n=\dfrac{d_n-y_{n-1}a_n}{b_n-u_{n-1}a_n}$。通过以上的消元过程,方程组(6.31)变为:

$$\begin{bmatrix}1 & u_1 & & & \\ & 1 & u_2 & & \\ & & \ddots & \ddots & \\ & & & 1 & u_{n-1} \\ & & & & 1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\\vdots\\x_{n-1}\\x_n\end{bmatrix}=\begin{bmatrix}y_1\\y_2\\\vdots\\y_{n-1}\\y_n\end{bmatrix}\tag{6.32}$$

其中,系数为:

$$\begin{cases}u_1=c_1/b_1\\ y_1=d_1/b_1\\ u_i=c_i/(b_i-u_{i-1}a_i),i=2,3,\cdots,n-1\\ y_i=(d_i-y_{i-1}a_i)/(b_i-u_{i-1}a_i),i=2,3,\cdots,n\end{cases}\tag{6.33}$$

2. 回代过程

方程组(6.32)实际上是一组递推关系式,自下而上回代,即可依次求出方程组的解:

$$\begin{cases}x_n=y_n\\ x_i=y_i-u_ix_{i+1},i=n-1,n-2,\cdots,1\end{cases}\tag{6.34}$$

以上就是解三对角方程组的追赶法。简言之,追赶法由如下两个过程构成:

(1) 追的过程(消元):按(6.33)式顺序逐步计算系数 $u_1,u_2,\cdots,u_{n-1}$ 和 $y_1,y_2,\cdots,y_n$。

(2) 赶的过程(回代):按(6.34)式逆序逐步求出解 $x_n,x_{n-1},\cdots,x_1$。

在追赶法的计算过程中不会出现小主元素,因此不会引起舍入误差的严重增长。虽然追赶

法的原理与Guass消去法相同，但考虑到方程组(6.31)的特点，计算时将系数中大量零元素撇开，从而大大节省了计算量(仅为$5n$次乘除法)。必须注意：为使追赶法的计算过程不致中断，必须保证(6.31)式的分母全不为零。

6.3　范数和误差分析

6.3.1　向量范数和矩阵范数

设$\boldsymbol{R}^n$为实n维向量空间，$\boldsymbol{C}^n$为复n维向量空间。用$\boldsymbol{K}^n$表示$\boldsymbol{R}^n$和$\boldsymbol{C}^n$，$\boldsymbol{K}^{n\times n}$表示$\boldsymbol{R}^{n\times n}$和$\boldsymbol{C}^{n\times n}$，$\boldsymbol{K}$为实数域$\boldsymbol{R}$或复数域$\boldsymbol{C}$。

1. 向量范数

定义6.2　若$\boldsymbol{K}^n$上任意一向量$\boldsymbol{x}$，对应一个非负实数$\|\boldsymbol{x}\|$，对于任意$\boldsymbol{y}\in\boldsymbol{K}^n$及$\alpha\in\boldsymbol{K}$，满足如下条件：

(1) 非负性　$\|\boldsymbol{x}\|>0$，且$\|\boldsymbol{x}\|=0$的充要条件是$\boldsymbol{x}=0$；

(2) 齐次性　$\|\alpha\boldsymbol{x}\|=|\alpha|\,\|\boldsymbol{x}\|$；

(3) 三角不等式　$\|\boldsymbol{x}+\boldsymbol{y}\|\leqslant\|\boldsymbol{x}\|+\|\boldsymbol{y}\|$。

则称$\|\boldsymbol{x}\|$为向量$\boldsymbol{x}$的范数。

常用的向量范数有：

(1) 1范数：$\|\boldsymbol{x}\|_1=\sum_{i=1}^{n}|x_i|$　　(2) 2范数：$\|\boldsymbol{x}\|_2=\left(\sum_{i=1}^{n}|x_i|^2\right)^{\frac{1}{2}}$

(3) ∞范数：$\|\boldsymbol{x}\|_\infty=\max_{1\leqslant i\leqslant n}|x_i|$　　(4) p范数：$\|\boldsymbol{x}\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{\frac{1}{p}}$

2. 矩阵范数

定义6.3　若$\boldsymbol{K}^{n\times n}$上任一矩阵$\boldsymbol{A}=(a_{ij})_{n\times n}$，对应一个非负实数$\|\boldsymbol{A}\|$，对于任意的$\boldsymbol{B}\in\boldsymbol{K}^{n\times n}$和$\alpha\in\boldsymbol{K}$，满足如下条件：

(1) 非负性　$\|\boldsymbol{A}\|>0$，且$\|\boldsymbol{A}\|=0$的充要条件是$\boldsymbol{A}=0$；

(2) 齐次性　$\|\alpha\boldsymbol{A}\|=|\alpha|\,\|\boldsymbol{A}\|$；

(3) 三角不等式　$\|\boldsymbol{A}+\boldsymbol{B}\|\leqslant\|\boldsymbol{A}\|+\|\boldsymbol{B}\|$；

(4) 乘法不等式　$\|\boldsymbol{AB}\|\leqslant\|\boldsymbol{A}\|\,\|\boldsymbol{B}\|$。

则称$\|\boldsymbol{A}\|$为矩阵$\boldsymbol{A}$的范数。

定义6.4　若向量$\boldsymbol{x}$的范数$\|\boldsymbol{x}\|$和矩阵$\boldsymbol{A}$的范数$\|\boldsymbol{A}\|$满足：

$$\|\boldsymbol{Ax}\|\leqslant\|\boldsymbol{A}\|\,\|\boldsymbol{x}\|$$

则称矩阵范数$\|\boldsymbol{A}\|$和向量范数$\|\boldsymbol{x}\|$是相容的。

定义6.5　设$\|\cdot\|$为$\boldsymbol{K}^n$上任一种向量范数，称

$$\|\boldsymbol{A}\|=\max_{\|\boldsymbol{x}\|=1}\|\boldsymbol{Ax}\|=\max_{\|\boldsymbol{x}\|\neq 0}\frac{\|\boldsymbol{Ax}\|}{\|\boldsymbol{x}\|}$$

为矩阵$\boldsymbol{A}$的范数，或由向量范数产生的从属范数或算子范数。

矩阵$\boldsymbol{A}$的几种从属范数有：

(1) 1 范数：$\| A \|_1 = \max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{n} |a_{ij}|$　　(2) ∞范数：$\| A \|_\infty = \max\limits_{1\leqslant i\leqslant n} \sum\limits_{j=1}^{n} |a_{ij}|$

(3) F 范数：$\| A \|_F = \left(\sum\limits_{i=1}^{n} \sum\limits_{j=1}^{n} |a_{ij}|^2 \right)^{\frac{1}{2}}$　　(4) 2 范数：$\| A \|_2 = \sqrt{\rho(A^H A)}$

在矩阵 A 的 2 范数中，称 $\rho(A^H A) = \max\limits_{1\leqslant i\leqslant n} |\lambda_i|$ 为 $A^H A$ 的谱半径，λ_i 为 $A^H A$ 的特征值，A^H 为 A 的共轭转置。在一般情况下，矩阵 A 的谱半径不超过 A 的任一范数，即：

$$\rho(A) < \| A \|$$

当 A 为正规矩阵即 $A^H A = AA^H$ 时，有：

$$\| A \|_2 = \rho(A)$$

3. 范数的有关定理

定理 6.3　设 $f(x) = \| x \|$ 为 $\boldsymbol{R}^n$ 上的任意向量范数，则 $f(x)$ 是 x 的连续函数。

定理 6.4　设 $\| x \|$ 和 $\| x \|_t$ 为向量 x 在 $\boldsymbol{R}^n$ 上的任意两种范数，则存在常数 $0 < c_1 \leqslant c_2$，使得下式成立：

$$c_1 \| x \| \leqslant \| x \|_t \leqslant c_2 \| x \|$$

定理 6.5　向量序列 $\{x^{(k)}\}$ 收敛于向量 x^* 的充要条件是：

$$\| x^{(k)} - x^* \| \to 0, k \to \infty$$

其中，$\| \cdot \|$ 为 $\boldsymbol{K}^n$ 上任一种向量范数。

证明　必要性：因为 $\lim\limits_{k\to\infty} x^{(k)} = x^*$，所以 $\lim\limits_{k\to\infty} (x^{(k)} - x^*) = 0$，故 $\lim\limits_{k\to\infty} \| x^{(k)} - x^* \| = 0$。

充分性：对于 $\boldsymbol{K}^n$ 上任一种向量范数 $\| x \|$，存在常数 $0 < c_1 \leqslant c_2$，使得

$$c_1 \| x^{(k)} - x^* \| \leqslant \| x^{(k)} - x^* \|_\infty \leqslant c_2 \| x^{(k)} - x^* \|$$

因为 $\lim\limits_{k\to\infty} \| x^{(k)} - x^* \| = 0$，所以 $\lim\limits_{k\to\infty} \| x^{(k)} - x^* \|_\infty = 0$，故

$$\lim_{k\to\infty} (x^{(k)} - x^*) = 0, \quad 从而 \quad \lim_{k\to\infty} x^{(k)} = x^*$$

定理 6.6　若 $A \in \boldsymbol{K}^{n\times n}$，则 $\lim\limits_{m\to\infty} A^m = 0$ 的充要条件是 $\rho(A) < 1$。

定理 6.7　若 $\| A \| < 1$，则 $I + A$ 为非奇异矩阵，且

$$\| (I+A)^{-1} \| \leqslant \frac{1}{1 - \| A \|}$$

证明　由 $\rho(A) \leqslant \| A \| < 1$ 可知，$I - A$ 非奇异，且 $\lim\limits_{m\to\infty} A^m = 0$。

又因为 $(I-A)(I+A+A^2+\cdots+A^k) = I - A^{k+1}$，所以

$$I + A + A^2 + \cdots + A^k = (I-A)^{-1}(I - A^{k+1})$$

当 $k\to\infty$ 时，得

$$\sum_{k=0}^{\infty} A^k = (I-A)^{-1},$$

$$\| (I-A)^{-1} \| \leqslant \sum_{k=0}^{\infty} \| A^k \| \leqslant \sum_{k=0}^{\infty} \| A \|^k = \frac{1}{1 - \| A \|}。$$

例 6.8　设 $A = \begin{bmatrix} 1 & -3 \\ -2 & 4 \end{bmatrix}$，计算 $\| A \|_1, \| A \|_\infty, \| A \|_F, \| A \|_2$。

解　$\| A \|_1 = \max(3, 7) = 7$　　$\| A \|_\infty = \max(4, 6) = 6$

$\| A \|_F = (1^2 + 2^2 + 3^2 + 4^2)^{\frac{1}{2}} = \sqrt{30}$

$A^H A = \begin{bmatrix} 5 & -11 \\ -11 & 25 \end{bmatrix}$，特征方程 $\lambda^2 - 30\lambda + 4 = 0$，$\lambda_{1,2} = 15 \pm \sqrt{221}$

$$\| A \|_2 = \sqrt{\rho(A^H A)} = \sqrt{15 + \sqrt{221}}$$

6.3.2 矩阵的条件数和误差分析

对于线性方程组 $AX=B$,A 为非奇异矩阵,X 为方程组的准确解。现分析各种扰动的方程组解的影响。

1. 设 A 精确,B 有小扰动 δB,相应地方程组的解为 $X+\delta X$

扰动方程推导如下:

$$A(X+\delta X) = B+\delta B$$

于是

$$\delta X = A^{-1}\delta B$$

$$\| \delta B \| \leqslant \| A^{-1} \| \ \| \delta B \|$$

由 $AX=B$,得:

$$\| B \| \leqslant \| A \| \ \| X \|$$

从而

$$\frac{\| \delta X \|}{\| X \|} \leqslant \| A^{-1} \| \ \| A \| \frac{\| \delta B \|}{\| B \|} \tag{6.35}$$

(6.35)式表明,相对误差 $\| \delta X \| / \| X \|$ 被放大了 $\| A^{-1} \| \ \| A \|$ 倍。由此可见,$\| A^{-1} \| \ \| A \|$ 是决定解好坏的主要因素之一,能够刻画解对原始数据扰动的灵敏程度。

定义 6.6 称 $\mathrm{cond}(A) = \| A^{-1} \| \ \| A \|$ 为矩阵 A 的条件数。通常使用的条件数有:

(1) $\mathrm{cond}(A)_\infty = \| A^{-1} \|_\infty \| A \|_\infty$;

(2) $\mathrm{cond}(A)_2 = \| A^{-1} \|_2 \| A \|_2 = \sqrt{\dfrac{\lambda_{\max}(A^T A)}{\lambda_{\min}(A^T A)}}$。当 A 为对称矩阵时,$\mathrm{cond}(A)_2 = \sqrt{\dfrac{\lambda_{\max}}{\lambda_{\min}}}$

其中,$\lambda_{\max}$,$\lambda_{\min}$ 分别是 A 的绝对值最大和最小的特征值。

当 $\mathrm{cond}(A) \gg 1$ 时,方程组是病态的,当 $\mathrm{cond}(A)$ 较小时,方程组是良态的。

2. 设 B 精确,A 有小扰动 δA,相应地方程组的解为 $X+\delta X$

扰动方程推导如下:

$$(A+\delta A)(X+\delta X) = B$$

$$(A+\delta A)X + (A+\delta A)\delta X = B$$

$$(A+\delta A)\delta X = -\delta A X$$

$$A(I+A^{-1}\delta A)\delta X = -\delta A X$$

得

$$\delta X = -(I+A^{-1}\delta A)^{-1} A^{-1} \delta A X$$

$$\frac{\| \delta X \|}{\| X \|} \leqslant \| (I+A^{-1}\delta A)^{-1} \| \ \| A^{-1} \| \ \| \delta A \|$$

由于 $A+\delta A$ 非奇异的充分条件是 $\| A^{-1}\delta A \| < 1$,故有:

$$\| (I+A^{-1}\delta A)^{-1} \| = \frac{1}{1 - \| A^{-1}\delta A \|},$$

而

$$\| A^{-1}\delta A \| \leqslant \| A^{-1} \| \ \| \delta A \| = \| A \| \ \| A^{-1} \| \frac{\| \delta A \|}{\| A \|} = \mathrm{cond}(A) \frac{\| \delta A \|}{\| A \|},$$

所以

$$\frac{\| \delta X \|}{\| X \|} \leqslant \| (I+A^{-1}\delta A)^{-1} \| \ \| A^{-1} \| \ \| \delta A \| = \frac{\| A^{-1} \| \ \| \delta A \|}{1 - \| A^{-1}\delta A \|}$$

$$\leqslant \frac{\| A^{-1} \| \| \delta A \|}{1 - \| A^{-1} \| \ \| \delta A \|}$$

即
$$\frac{\|\delta X\|}{\|X\|}\leqslant\frac{\operatorname{cond}(A)\frac{\|\delta A\|}{\|A\|}}{1-\operatorname{cond}(A)\frac{\|\delta A\|}{\|A\|}} \tag{6.36}$$

从上面的分析中可以看出,条件数cond(A)是判断一个矩阵是否为病态的一个重要参量。当cond(A)=1时,方程组的状态最好。

6.4　解线性方程组的迭代法

解线性方程组的迭代法是将求解方程组的问题转化为一个无穷序列,用这个无穷序列去逐步逼近精确解。即:

$$AX=B \iff X=GX+f \iff X^{(k+1)}=GX^{(k)}+f$$

这种方法算法简单,编程较容易,但对方程组的系数矩阵A有特殊要求,以便保证迭代过程收敛。常用于解高阶方程组和系数矩阵是稀疏矩阵的线性方程组。

6.4.1　Jacobi(雅可比)迭代法

对于线性方程组(6.1),若$a_{ii}\neq0(i=1,2,\cdots,n)$,则可以改写为:

$$\begin{cases}x_1=1/a_{11}(\quad -a_{12}x_2-\cdots-a_{1n}x_n+b_1)\\ x_2=1/a_{22}(-a_{21}x_1 \quad -\cdots-a_{2n}x_n+b_2)\\ \quad\cdots\\ x_n=1/a_{nn}(-a_{n1}x_1-\cdots-a_{nn-1}x_{n-1} \quad +b_n)\end{cases} \tag{6.37}$$

由此可以列出迭代格式为:

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\Big(b_i-\sum_{\substack{j=1\\ j\neq i}}^{n}a_{ij}x_j^{(k)}\Big) \qquad (i=1,2,\cdots,n)$$

或
$$x_i^{(k+1)}=\frac{1}{a_{ii}}\Big(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\Big) \qquad (i=1,2,\cdots,n) \tag{6.38}$$

称(6.38)为Jacobi迭代法。下面用矩阵形式来表示Jacobi迭代法。

对线性方程组$AX=B$,假设系数矩阵A非奇异,并且对角元素$a_{ii}\neq0(i=1,2,\cdots,n)$,将系数矩阵$A$作如下分解:

$$A=L+D+U$$

其中

$$L=\begin{bmatrix}0&&&\\a_{21}&0&&\\ \vdots&\vdots&0&\\a_{n1}&a_{n2}&\cdots&0\end{bmatrix},\quad D=\begin{bmatrix}a_{11}&&&\\&a_{22}&&\\&&\ddots&\\&&&a_{nn}\end{bmatrix},\quad U=\begin{bmatrix}0&a_{12}&\cdots&a_{1n}\\&0&\cdots&a_{2n}\\ \vdots&\vdots&0&\vdots\\&&\cdots&0\end{bmatrix}$$

于是有:

$$(L+D+U)X=B$$
$$DX=B-(L+U)X$$
$$X=-D^{-1}(L+U)X+D^{-1}B$$

得到Jacobi迭代公式为:

$$X^{(k+1)}=-D^{-1}(L+U)X^{(k)}+D^{-1}B$$

令$B_J=-D^{-1}(L+U)$，$f=D^{-1}B$，得：

$$X^{(k+1)}=B_J X^{(k)}+f \tag{6.39}$$

称矩阵B_J为Jacobi迭代法的迭代矩阵。

应用Jacobi迭代法解线性方程组，先给出各个变量$x_1,x_2,\cdots,x_n$的初始迭代值$X^{(0)}$，按迭代格式进行计算，将第k次的迭代计算结果$X^{(k)}$与第$k-1$次的迭代计算结果$X^{(k-1)}$进行比较，若满足$|X^{(k)}-X^{(k-1)}|<\varepsilon$，则终止迭代过程，得到迭代计算结果$X^{(k)}$；否则，继续进行迭代，直至满足精度要求$\varepsilon$为止。

例6.9　用Jacobi迭代法解线性方程组：

$$\begin{cases}10x_1-x_2-2x_3=7.2\\-x_1+10x_2-2x_3=8.3\\-x_1-x_2+5x_3=4.2\end{cases}$$

解　建立迭代格式：

$$\begin{cases}x_1^{(k+1)}=0.1x_2^{(k)}+0.2x_3^{(k)}+0.72\\x_2^{(k+1)}=0.1x_1^{(k)}+0.2x_3^{(k)}+0.83\\x_3^{(k+1)}=0.2x_1^{(k)}+0.2x_2^{(k)}+0.84\end{cases}$$

给出迭代初值：$x_1^{(0)}=x_2^{(0)}=x_3^{(0)}=0$，计算结果见表6.3。

表6.3　例6.9迭代计算结果

k	0	1	2	3	4	5	6	7	8
$x_1^{(k)}$	0.00000	0.72000	0.97100	1.05700	1.08535	1.09510	1.09834	1.09944	1.09981
$x_2^{(k)}$	0.00000	0.83000	1.07000	1.15710	1.18534	1.19510	1.19834	1.19981	1.19941
$x_3^{(k)}$	0.00000	0.84000	1.15000	1.24820	1.28282	1.29414	1.29504	1.29934	1.29978

迭代进行8，9次(k=8，9)的结果为：

$$x_1^{(8)}=1.09981,\quad x_2^{(8)}=1.19941,\quad x_3^{(8)}=1.29978$$

$$x_1^{(9)}=1.09994,\quad x_2^{(9)}=1.19994,\quad x_3^{(9)}=1.29992$$

当迭代次数再增加时，迭代结果将越来越接近方程组的精确解，即：

$$x_1=1.1,\quad x_2=1.2,\quad x_3=1.3$$

例6.10　用Jacobi迭代法(矩阵形式)解下列方程组(精确到10^{-3})：

$$\begin{bmatrix}4&0.24&-0.08\\0.09&3&-0.15\\0.04&-0.08&4\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}8\\9\\20\end{bmatrix}$$

解　将系数矩阵A的对角元素归一化，得：

$$\begin{bmatrix}1&0.06&-0.02\\0.03&1&-0.05\\0.01&-0.02&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}2\\3\\5\end{bmatrix}$$

Jacobi迭代公式为：

$$\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}=\begin{bmatrix}0&-0.06&0.02\\-0.03&0&0.05\\-0.01&0.02&0\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix}+\begin{bmatrix}2\\3\\5\end{bmatrix}$$

由于$\|G\|_\infty=0.08<1$，所以对于任意的初始向量$x_0\in R^3$。上述迭代过程产生的向量序列

均收敛。取 $x^{(0)}=[2,3,5]'$ 做迭代初值,有:

$$x^{(1)}=\begin{bmatrix}x_1^{(1)}\\x_2^{(1)}\\x_3^{(1)}\end{bmatrix}=\begin{bmatrix}0&-0.06&0.02\\-0.03&0&0.05\\-0.01&0.02&0\end{bmatrix}\begin{bmatrix}2\\3\\5\end{bmatrix}+\begin{bmatrix}2\\3\\5\end{bmatrix}=\begin{bmatrix}1.92\\3.19\\5.04\end{bmatrix}$$

$$x^{(2)}=\begin{bmatrix}x_1^{(2)}\\x_2^{(2)}\\x_3^{(2)}\end{bmatrix}=\begin{bmatrix}0&-0.06&0.02\\-0.03&0&0.05\\-0.01&0.02&0\end{bmatrix}\begin{bmatrix}1.92\\3.19\\5.04\end{bmatrix}+\begin{bmatrix}2\\3\\5\end{bmatrix}=\begin{bmatrix}1.9094\\3.1944\\5.0446\end{bmatrix}$$

$$x^{(3)}=\begin{bmatrix}x_1^{(3)}\\x_2^{(3)}\\x_3^{(3)}\end{bmatrix}=\begin{bmatrix}0&-0.06&0.02\\-0.03&0&0.05\\-0.01&0.02&0\end{bmatrix}\begin{bmatrix}1.9094\\3.1944\\5.0446\end{bmatrix}+\begin{bmatrix}2\\3\\5\end{bmatrix}=\begin{bmatrix}1.909228\\3.194948\\5.044794\end{bmatrix}$$

$$x^{(3)}-x^{(2)}=\begin{bmatrix}-0.000172\\0.000548\\0.000194\end{bmatrix}$$

$$\|x^{(3)}-x^{(2)}\|_\infty=0.000548$$

$$\|x^*-x^{(3)}\|_\infty\leqslant\frac{0.08}{1-0.08}\|x^{(3)}-x^{(2)}\|_\infty=\frac{0.08}{1-0.08}\times0.000548$$

$$<0.09\times0.000548=0.00004932<0.00005=0.5$$

因为 $x^{(3)}$ 已满足精度要求,故停止迭代计算。$x^{(3)}$ 即为满足精度要求的近似解。

6.4.2 Guass-Seidel(高斯 - 赛德尔)迭代法

在Jacobi迭代法的第 $k+1$ 次迭代过程中,计算所有的 $X^{(k+1)}$(即 $x_1^{(k+1)},x_2^{(k+1)},\cdots,x_n^{(k+1)}$)都是利用第 k 次迭代计算的结果 $X^{(k)}$ 进行的。事实上,在计算 $x_i^{(k+1)}$ 时 $x_1^{(k+1)},x_2^{(k+1)},\cdots,x_{i-1}^{(k+1)}$ 都已经计算出来了,如果方程组迭代过程是收敛的,那么最新计算出的 $x_1^{(k+1)},x_2^{(k+1)},\cdots,x_{i-1}^{(k+1)}$ 要比旧值 $x_1^{(k)},x_2^{(k)},\cdots,x_{i-1}^{(k)}$ 更加逼近精确解。因此,可以利用这些新值来代替旧值,加速迭代过程。于是,对Jacobi迭代法加以修改,可得到新的迭代公式:

$$x_i^{(k+1)}=\frac{1}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right)\qquad(i=1,2,\cdots,n)\tag{6.40}$$

称(6.40)为Guass-Seidel迭代法(GS法)。其矩阵形式是:

$$(L+D+U)X=B$$

$$DX=B-LX-UX$$

$$X^{(k+1)}=-D^{-1}LX^{(k+1)}-D^{-1}UX^{(k)}+D^{-1}B$$

如果 $(D+L)^{-1}$ 存在,则有:

$$(D+L)X^{(k+1)}=-UX^{(k)}+B$$

$$X^{(k+1)}=-(D+L)^{-1}UX^{(k)}+(D+L)^{-1}B$$

令 $B_G=-(D+L)^{-1}U,f=(D+L)^{-1}B$,得:

$$X^{(k+1)}=B_GX^{(k)}+f\tag{6.41}$$

称矩阵 B_G 为Guass-Seidel迭代法的迭代矩阵。

为了节省存储单元,可将新值 $x_i^{(k+1)}$ 存入旧值 $x_i^{(k)}$ 所占用的单元内,只需把(6.40)式括号中的后两项合并,即可写如下的动态形式:

$$\left(b_i-\sum_{\substack{j=1\\j\neq i}}^{n}a_{ij}x_j\right)\Big/a_{ii}\ \Rightarrow\ x_i\qquad i=1,2,3,\cdots,n$$

一般来说，Guass-Seidel 迭代法比 Jacobi 迭代法好，但有时可能出现 Guass-Seidel 迭代法收敛慢，甚至发散的情况，而Jacobi 迭代法反而收敛。使用时，要根据具体实际问题选择恰当的算法。

例 6.11 用Guass-Seidel 迭代法解线性方程组：

$$\begin{cases}10x_1-x_2-2x_3=7.2\\-x_1+10x_2-2x_3=8.3\\-x_1-x_2+5x_3=4.2\end{cases}$$

解 建立迭代格式，即：

$$\begin{cases}x_1^{(k+1)}=0.1x_2^{(k)}+0.2x_3^{(k)}+0.72\\x_2^{(k+1)}=0.1x_1^{(k+1)}+0.2x_3^{(k)}+0.83\\x_3^{(k+1)}=0.2x_1^{(k+1)}+0.2x_2^{(k+1)}+0.84\end{cases}$$

给出迭代初值：$x_1^{(0)}=x_2^{(0)}=x_3^{(0)}=0$，计算结果见表 6.4。

表 6.4 例 6.11 的迭代计算结果

k	0	1	2	3	4	5	6
$x_1^{(k)}$	0.00000	0.72000	1.04308	1.09313	1.09913	1.09989	1.09999
$x_2^{(k)}$	0.00000	0.90200	1.16719	1.19572	1.19947	1.19993	1.19999
$x_3^{(k)}$	0.00000	1.16440	1.28205	1.29778	1.29972	1.29996	1.30000

6.4.3 超松弛迭代法

超松弛法是对Guass-Seidel 迭代法的进一步改进，它是一种线性加速方法，可以加快迭代的收敛速度。其基本思想是，将前一步迭代结果 $x_i^{(k)}$ 与 Guass-Seidel 迭代法所求得的迭代值 $x_i^{(k+1)}$ 适当加权平均，以获得更好的近似值 $x_i^{(k+1)}$。即：

$$x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\frac{\omega}{a_{ii}}\left(b_i-\sum_{j=1}^{i-1}a_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{n}a_{ij}x_j^{(k)}\right),\qquad i=1,2,\cdots,n \tag{6.42}$$

(6.42)式称为超松弛迭代法(SOR)，其中 ω 是松弛因子。当松弛因子 $\omega=1$ 时，超松弛法(6.42)式就简化为 Guass-Seidel 迭代法。

可以证明，对于线性方程组 $AX=B$，有：

(1) 超松弛迭代法收敛的必要条件是 $0<\omega<2$；

(2) 若 A 为对称正定阵，且 $0<\omega<2$，则超松弛迭代过程收敛；

(3) 若 A 为严格对角占优阵，则当 $0<\omega<1$ 时，迭代过程收敛。

当超松弛迭代法收敛时，如何选取 ω 使收敛速度最快呢？目前只有少数特殊类型的矩阵，才有计算 ω 的最佳值的理论公式。因此，在实际应用时有一定困难。通常是采用试算法来确定 ω 的最佳近似值，或者是先取一个 ω 值，根据迭代过程收敛速度的快慢，随时修改 ω，从而逐步找到最佳的 ω 值。

例 6.12 试用超松弛迭代法求图 6.1 所示电路网络中节点 a，b 和 c 处的电压。

解 根据电路理论，流入和流出节点 a、b、c 处的电流总和为零，即：

$$(e_a-20)/2+(e_a-e_b)/4+(e_a-e_c)/3=0$$

$$(e_b-e_a)/4+(e_b-0)/3+(e_b-e_c)/5=0$$

$$(e_c-5)/3+(e_c-e_a)/3+(e_c-e_b)/5=0$$

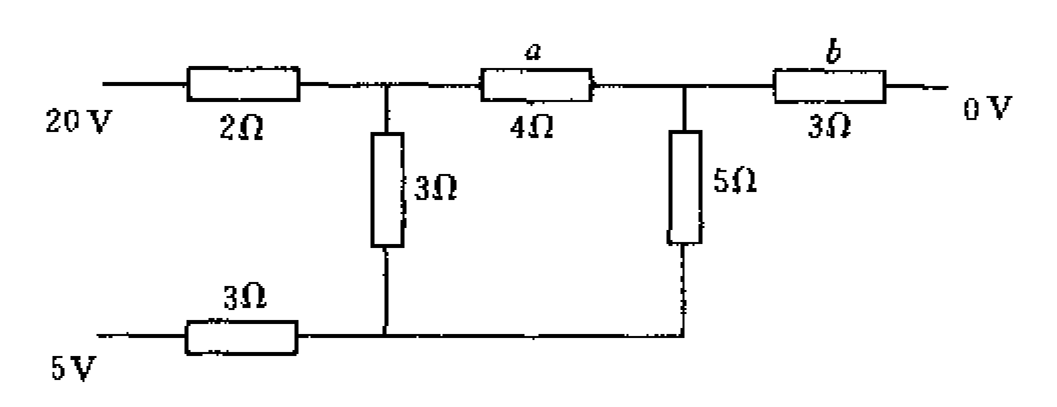


图 6.1　电路网络图

改写为等价方程组，即：

$$(1/2+1/4+1/3)e_a-e_b/4-e_c/3=20/2$$
$$-e_a/4+(1/4+1/3+1/5)e_b-e_c/5=0$$
$$-e_a/3-e_b/5+(1/3+1/3+1/5)e_c=5/3$$

用MATLAB 语言求解方程组的程序如下：

```
clear
a=[1/2 +1/4+1/3,-1/4,-1/3;
        -1/4,1/4+1/3+1/5,-1/5;
        -1/3,-1/5,1/3+1/3+1/5],
y=[20/2,0,5/3];
x=zeros(1,3);
w=1.2;
for it=1:50
    error=0;
    for i=1:3
        s=0;
        xb=x(i);
        for j=1:3
            if(i~=j)
                s=s+a(i,j) * x(j);
            end
        end
        x(i)=w * (y(i)-s)/a(i,i)+(1-w) * x(i);
        error=error+abs(x(i)-xb);
    end
    fprintf(' It.No=%3.0f,error=%7.2e\n',it,error)
    if error/3<0.0001
        break
    end
end
x
```

程序运行结果如下：

```
It. No. =     1,error=2.39e + 01
It. No. =     2,error=4.75e + 00
It. No. =     3,error=7.24e - 01
It. No. =     4,error=2.64e - 01
It. No. =     5,error=5.03e - 02
It. No. =     6,error=1.36e - 02
It. No. =     7,error=4.39e - 03
It. No. =     8,error=1.16e - 03
It. No. =     9,error=3.03e - 04
It. No. =    10,error=8.55e - 05
x=     13.3453     6.4401     8.5420
```

6.4.4 迭代法的收敛性

定义6.7　如果对固定的矩阵B及向量f,对任意的初始向量$X^{(0)}$,由迭代公式

$$X^{(k+1)}=BX^{(k)}+f \tag{6.43}$$

得出的向量序列$X^{(k)}$都有

$$\lim_{k\to\infty}X^{(k)}=X^*$$

成立,其中X^*是一确定向量,不依赖于$X^{(k)}$的选取。则称迭代公式(6.43)是收敛的,否则是发散的。

判断解方程组迭代法的收敛定理如下:

定理6.8　设有方程组$X=BX+f$,对于任意的初始向量$X^{(0)}$及任意的f,迭代公式$X^{(k+1)}=BX^{(k)}+f$收敛的充要条件是:

$$\rho(B)<1$$

证明　假设X^*为方程组$X=BX+f$的准确解,即:

$$X^*=BX^*+f$$

对于任意的初值$X^{(0)}$和任意的f,迭代公式为:

$$X^{(k+1)}=BX^{(k)}+f$$

于是

$$\begin{aligned}X^{(k)}-X^*&=BX^{(k-1)}+f-(BX^*+f)\\&=B(X^{(k-1)}-X^*)\\&=B^2(X^{(k-2)}-X^*)\\&\;\;\vdots\\&=B^k(X^{(0)}-X^*)\end{aligned}$$

因为$\lim\limits_{k\to\infty}B^k=0$的充要条件是$\rho(B)<1$,所以$\lim\limits_{k\to\infty}X^{(k)}=X^*$充要条件是$\rho(B)<1$。

例6.13　考察迭代公式$X^{(k+1)}=BX^{(k)}+f$的收敛性。其中矩阵B为:

$$B=\begin{bmatrix}0 & \frac{3}{8} & -\frac{2}{8}\\ -\frac{4}{11} & 0 & \frac{1}{11}\\ -\frac{6}{12} & -\frac{3}{12} & 0\end{bmatrix}$$

解 计算矩阵 B 的特征值,即:

$$\det(\lambda E-B)=\lambda^3+0.034090909\lambda+0.039772727=0$$

$$\lambda_1=-0.3082,\quad \lambda_2=0.1541+0.3245i,\quad \lambda_3=0.1541-0.3245i$$

$$|\lambda_1|=0.3082<1,\quad |\lambda_2|=|\lambda_3|=0.3592<1$$

$\rho(B)<1$,故 $X^{(k+1)}=BX^{(k)}+f$,对于任意 $X^{(0)}$ 均收敛。

定义 6.8 若 n 阶矩阵 A 满足:

$$\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|<|a_{ii}|\qquad i=1,2,\cdots,n$$

即矩阵 A 主对角线元素的绝对值大于同一行中其他元素绝对值之和,则称矩阵 A 为严格对角占优矩阵。

定理 6.9 若 A 为严格对角占优矩阵,则解方程组 $AX=B$ 的 Jacobi 迭代法、Guass-Seidel 迭代法均收敛,对于 SOR 方法,当 $0<\omega<1$ 时迭代收敛。

在实际应用中,用该定理判断迭代法的收敛性是容易的。

6.5 非线性方程组的数值解法

非线性方程组不能用直接法来求解,只能用迭代方法,且最常用的是牛顿迭代法。前面已经讲过的牛顿法是把非线性问题化为线性问题来求解,这种方法也可用于解非线性方程组的问题,即设法将一个非线性方程组化为线性方程组来求解。这种转化是基于多元函数的泰勒展开。

非线性方程最一般的形式为:

$$\begin{cases}f_1(x_1,x_2,\cdots,x_n)=0\\ f_2(x_1,x_2,\cdots,x_n)=0\\ \quad\cdots\\ f_n(x_1,x_2,\cdots,x_n)=0\end{cases}\tag{6.44}$$

假设该非线性方程组的初始近似解为 $(x_1,x_2,\cdots,x_n)$,和方程组精确解 $(x_1^*,x_2^*,\cdots,x_n^*)$ 的误差为 $(\Delta x_1,\Delta x_2,\cdots,\Delta x_n)$。将(6.44)在点 $(x_1,x_2,\cdots,x_n)$ 处进行 Taylor 展开,并忽略高阶项,得:

$$\begin{cases}f_1(x_1+\Delta x_1,\cdots,x_n+\Delta x_n)=f_1(x_1,x_2,\cdots,x_n)+\dfrac{\partial f_1}{\partial x_1}\cdot\Delta x_1+\cdots+\dfrac{\partial f_1}{\partial x_n}\cdot\Delta x_n\\ \quad\cdots\\ f_n(x_1+\Delta x_1,\cdots,x_n+\Delta x_n)=f_n(x_1,x_2,\cdots,x_n)+\dfrac{\partial f_n}{\partial x_1}\cdot\Delta x_1+\cdots+\dfrac{\partial f_n}{\partial x_n}\cdot\Delta x_n\end{cases}\tag{6.45}$$

在(6.45)式左边,$x_i+\Delta x_i$ 即为方程组(6.44)的根 $(x_1^*,x_2^*,\cdots,x_n^*)$,所以

$$f_k(x_1^*,x_2^*,\cdots,x_n^*)=0\qquad(k=1,2,\cdots,n)$$

故

$$\sum_i\frac{\partial f_k}{\partial x_i}\Delta x_i=-f_k(x_1,x_2,\cdots,x_n)\qquad(k=1,2,\cdots,n)\tag{6.46}$$

其中,偏导数是用初值计算的,等式(6.46)的右边成了常数矩阵。将上式写成矩阵形式:

$$J\Delta x=-f$$

(6.46)式是关于$\Delta x_i(i=1,2,\cdots,n)$的线性方程组。当系数矩阵的行列式$J\neq0$时，方程组有惟一解，$J$称为Jacobian(雅克比)矩阵。这样，就把求解非线性方程组问题化为求解线性方程组的根Δx_i的问题了。用它作为修正因子，则得到方程组(6.44)的近似解

$$x_i^{(K+1)}=x_i^{(K)}+\Delta x_i^{(K)} \qquad i=1,2,\cdots,n \tag{6.47}$$

式中，$K=0,1,2,\cdots$为迭代次数。当前后两次迭代结果

$$|x_i^{(K+1)}-x_i^{(K)}|\leqslant\varepsilon$$

时，迭代过程结束；否则，继续迭代直到满足精度要求为止。在迭代过程中，常用$\frac{\Delta x_i}{x_i}\leqslant0.00001$ $(i=1,2,\cdots,n)$作为检验修正因子Δx_i是否充分小的标准。开始迭代时，给出迭代初值$x_i^{(0)}$。

牛顿迭代法是较为满意的求解线性方程组的方法，它同样存在收敛性问题。一般牛顿迭代法的收敛域的大小与方程组中方程的阶数和方程的个数成相反的关系。牛顿迭代法对初值的要求较高，当选取的初值数为精确时，收敛速度是很快的(具有二阶收敛速度)。但每迭代一次都要计算一次系数矩阵，解一次线性方程组，因此计算量也是相当大的，幸好一般工程问题的方程阶数和个数都不会太高。

例6.14　用牛顿迭代法解：

$$4x_1-x_2+\frac{1}{10}e^{x_1-1}=0$$

$$-x_1+4x_2+\frac{1}{8}x_1^2=0$$

解　计算方程组左边函数表达式的偏导数以及(6.46)式右边的常数f_i，即：

$$\frac{\partial f_1}{\partial x_1}=4+\frac{1}{10}e^{x_1},\quad \frac{\partial f_1}{\partial x_2}=-1$$

$$\frac{\partial f_2}{\partial x_1}=-1+\frac{1}{4}x_1,\quad \frac{\partial f_2}{\partial x_2}=4$$

取迭代初值为$x_1^{(0)}=x_2^{(0)}=0$，则

$$\frac{\partial f_1}{\partial x_1}=4.1,\quad \frac{\partial f_1}{\partial x_2}=-1,\quad \frac{\partial f_2}{\partial x_1}=-1,\quad \frac{\partial f_2}{\partial x_2}=4$$

$$f_1(0)=\frac{9}{10},\quad f_2(0)=0$$

按(6.46)式，则可以列出关于Δx_1和Δx_2的线性方程组：

$$4.1\Delta x_1^{(0)}-\Delta x_2^{(0)}=\frac{9}{10}$$

$$-\Delta x_1^{(0)}+\Delta x_2^{(0)}=0$$

解之，得：

$$\Delta x_1^{(0)}=0.233766233$$

$$\Delta x_2^{(0)}=0.058441555$$

和

$$x_1^{(1)}=x_1^{(0)}+\Delta x_1^{(0)}=0.233766233$$

$$x_2^{(1)}=x_2^{(0)}+\Delta x_2^{(0)}=0.058441555$$

继续迭代计算，直到满足精度要求为止。所得结果见表6.5。

表6.5 例6.14的求解结果

K	Δx_1	x_1	Δx_2	x_2
0		0		0
1	0.235766233	0.233766233	0.058441555	0.058441555
2	−0.00115999	0.232606243	−0.001828277	0.056613327
3	−0.00003923772	0.232567006	−0.000168072	0.056445255
4	-8.770872×10^{-9}	0.232566997	0.2628097×10^{-6}	0.056451517

非线性方程组的解为:

$$x_1 \cong 0.23257,\quad x_2 \cong 0.05645$$

本 章 小 结

本章主要讨论了线性方程组的数值解法。

迭代方法是将线性方程组转化为一种能够收敛的迭代格式。迭代格式的结构和迭代初值决定了迭代的可能性,以及结果的精度和迭代速度。雅可比迭代法、G-S方法和超松弛(SOR)法三种迭代方法之间是有联系的,后者是前者的改进。G-S法迭代过程的更新值立即可以被利用,从而有利于提高收敛速度;此外,G-S方法中,旧值可以被新值覆盖,所以程序结构很简单。SOR方法利用了一个松弛因子,使迭代速度比G-S方法更快。

线性方程组的消去法由消元和回代两个过程组成。选列主元消去法可以避免解题过程的中断。对于系数矩阵是三角对角阵(也是稀疏矩阵)的特殊线性方程组,则常常采用追赶法。

运用线性代数的数学原理解线性方程组更为简洁。可以利用矩阵的LU分解函数,做几步简单运算即可求得方程组的解。

解非线性方程组的牛顿迭代法基于泰勒级数,将非线性方程变为线性方程求解。

学习本章主要应掌握:

(1) 迭代格式的建立基础;

(2) 雅可比迭代法、G-S方法和超松弛(SOR)法三种迭代方法的共同点和不同点;

(3) 高斯选列主元消去法的优点;

(4) 追赶法应用场合;

(5) 各种矩阵分解法;

(6) 解非线性方程组的牛顿迭代法。

习 题 六

6.1 证明下列左边的线性方程组$AX=B$等价于右边的上三角线性方程组$UX=Y$,并解方程组。

(1) $\begin{cases}2x_1+4x_2-6x_3=-4\\ x_1+5x_2+3x_3=10\\ x_1+3x_2+2x_3=5\end{cases}$ $\qquad\begin{array}{l}2x_1+4x_2-6x_3=-4\\ 3x_2+6x_3=12\\ 3x_3=3\end{array}$

(2) $\begin{cases}x_1+x_2+6x_3=7\\ -x_1+2x_2+9x_3=2\\ x_1-2x_2+3x_3=10\end{cases}$ $\qquad\begin{array}{l}x_1+x_2+6x_3=7\\ 3x_2+15x_3=9\\ 12x_3=12\end{array}$

$$(3)\begin{cases}2x_1-2x_2+5x_3=6\\2x_1+3x_2+x_3=13\\-x_1+4x_2-4x_3=3\end{cases}\qquad\begin{array}{l}2x_1-2x_2+5x_3=6\\5x_2-4x_3=7\\0.9x_3=1.8\end{array}$$

$$(4)\begin{cases}-5x_1+2x_2-x_3=-1\\x_1+0x_2+3x_3=5\\3x_1+x_2+6x_3=17\end{cases}\qquad\begin{array}{l}-5x_1+2x_2-x_3=-1\\0.4x_2+2.8x_3=4.8\\-10x_3=-10\end{array}$$

$$(5)\begin{cases}4x_1+8x_2+4x_3+0x_4=8\\x_1+5x_2+4x_3-3x_4=-4\\x_1+4x_2+7x_3+2x_4=10\\x_1+3x_2+0x_3-2x_4=-4\end{cases}\qquad\begin{array}{l}4x_1+8x_2+4x_3+0x_4=8\\3x_2+3x_3-3x_4=-6\\4x_3+4x_4=12\\x_4=2\end{array}$$

$$(6)\begin{cases}x_1+2x_2+0x_3-x_4=9\\2x_1+3x_2-x_3+0x_4=9\\0x_1+4x_2+2x_3-5x_4=26\\5x_1+5x_2+2x_3-4x_4=32\end{cases}\qquad\begin{array}{l}2x_1+4x_2-4x_3+0x_4=12\\x_1+5x_2-5x_3-3x_4=18\\2x_1+3x_2+x_3+3x_4=8\\x_1+4x_2-2x_3+2x_4=8\end{array}$$

$$(7)\begin{cases}2x_1+4x_2-4x_3+0x_4=12\\3x_2-3x_3-3x_4=12\\4x_3+2x_4=0\\3x_4=-6\end{cases}\qquad\begin{array}{l}x_1+2x_2+0x_3-x_4=9\\-x_2-x_3+2x_4=-9\\-2x_3+3x_4=-10\\1.5x_4=-3\end{array}$$

6.2 试将下列矩阵分解为 L,U 矩阵:

$$A=\begin{pmatrix}1&2&6\\4&8&-1\\-2&3&5\end{pmatrix}$$

6.3 用高斯消去法解下列方程组:

$$(1)\begin{cases}2x_1+x_2+x_3=4\\x_1+3x_2+2x_3=6\\x_1+2x_2+2x_3=5\end{cases}$$

$$(2)\begin{cases}x_1+x_2-x_3=3\\2x_1+x_2-x_3=0\\-x_1-2x_2+2x_3=-5\end{cases}$$

$$(3)\begin{cases}3x_1-x_2+2x_3=-5\\x_1+x_2+x_3=-4\\2x_1+x_2-x_3=-3\end{cases}$$

6.4 用列主元消去法解下列方程组并求行列式:

$$(1)\begin{cases}2x_1-x_2-x_3=4\\3x_1+4x_2-2x_3=11\\3x_1-2x_2+4x_3=11\end{cases}$$

$$(2)\begin{cases}3x_1-x_2+4x_3=7\\-x_1+2x_2-2x_3=-1\\2x_1-3x_2-2x_3=0\end{cases}$$

6.5 用雅可比迭代法解下列方程组:

(1) $\begin{cases} 5x_1-2x_2+x_3=4 \\ x_1+5x_2-3x_3=2 \\ 2x_1+x_2-5x_3=-11 \end{cases}$

(2) $\begin{cases} 2x_1+x_2=1 \\ x_1-4x_2=5 \end{cases}$

(3) $\begin{cases} 2x_1 & =3 \\ x_1+1.5x_2 & =4.5 \\ -3x_1+0.5x_3 & =-6.6 \\ 2x_1-2x_2+x_3+x_4 & =0.8 \end{cases}$

6.6　用G-S迭代法解题6.5。

6.7　用超松弛法解题6.5，$\omega=1.2$。

6.8　图6.2给出了某城市单行街道的交通流量(每小时过车数)网络图。假设：

(1) 网络的全部流入车量等于全部流出车量；

(2) 每个节点的流入车量等于流出车量。

试建立数学模型确定该交通网络未知部分的具体流量。

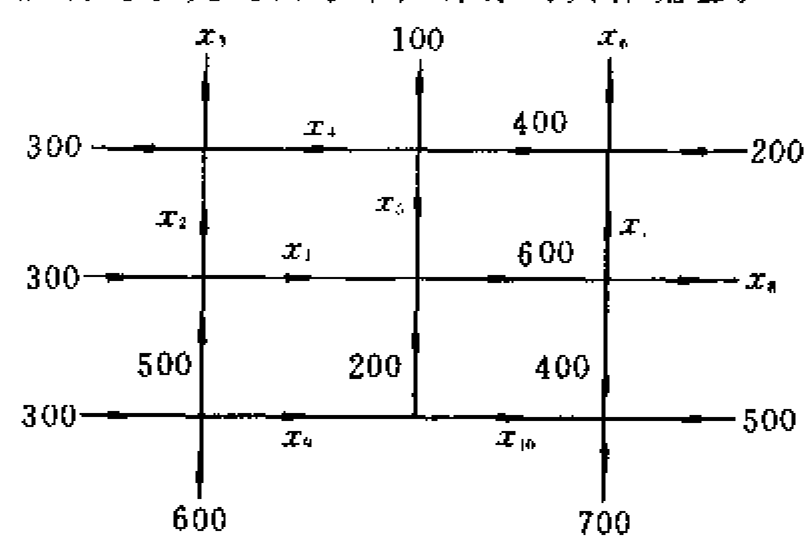


图6.2　某市交通流量网络图

6.9　设有线性方程组：

$$\begin{cases} 5x_1+2x_2+x_3=-12 \\ -x_1+4x_2+2x_3=20 \\ 2x_1-3x_2+10x_3=3 \end{cases}$$

取初始向量 $x^{(0)}=[-3,1,1]^T$，分别用雅可比迭代和G-S迭代法求解，要求满足 $\max|x_i^{(k+1)}-x_i^{(k)}|\leqslant 10^{-3}$ 时终止迭代。

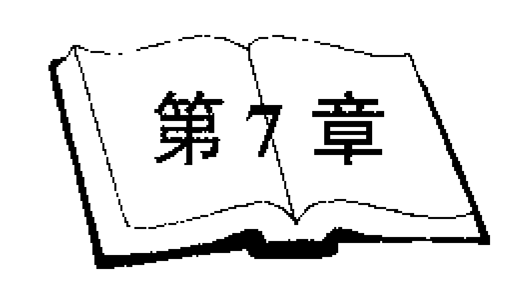
6.10　用MATLAB程序写出M文件，对下列矩阵进行选列主元消去法计算逆矩阵：

$$A=\begin{bmatrix} 0 & 5 & 1 \\ -1 & 6 & 3 \\ 3 & -9 & 5 \end{bmatrix}$$

6.11　用LU分解求解下列方程：

(1) $\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$

(2) $\begin{bmatrix} 2 & -1 & 1 \\ -3 & 4 & -1 \\ 1 & -1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}=\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix}$





本章简明地介绍了MATLAB语言的基本知识和编程技术，供教学者使用或参考。

7.1 MATLAB的特点

1984年，MathWorks公司推出数值计算软件MATLAB，历经20多年的发展，MATLAB已经成为21世纪最流行的科学计算软件。MATLAB具有强大的科学计算能力和出色的图形处理功能，广泛地应用于教学和科研之中，是人们进行科学计算等工作的强大而有力的工具。概括起来，MATLAB具有以下优点：

1）MATLAB 功能强大

MATLAB分为总软件包和若干个工具箱（为许多专业领域开发了功能强大的模块集或工具箱，拥有600多个工程中要用到的数学运算函数），将数值计算功能、符号运算功能和图形处理功能高度地集成在一起，用户直接使用这些工具箱而不需要自己编写代码，就可以方便地实现数值分析、优化、统计、自动控制、信号处理、图像处理等若干个领域的科学计算、符号运算、数据可视化和图形、图像处理等任务。其中，强大的矩阵计算功能是MATLAB科学计算功能的重要组成部分，是区别其他程序语言的重要标志。

2）MATLAB 图形处理功能出色

MATLAB具有出色的科学计算可视化功能，不仅可以绘制一般的二维、三维图形，如线图、条形图、饼图、散点图、直方图与误差条图等，还可以绘制工程特性较强的特殊图形，如玫瑰花图、极坐标图等。在数据处理方面，MATLAB可以用丰富的图形来表现数据的特征，如二维三维等值线图、矢量图、三维表面图、假彩色图、云图、二维三维流线图、三维彩色流锥图、流沙图、流带图、流管图、卷曲图与切片图等。

3）MATLAB 程序代码简洁

MATLAB语法规则与一般的结构化高级编程语言如C语言等大同小异，但MATLAB实现了常量、变量、向量和矩阵等数据类型的高度统一，用户不需要事先分别定义数据类型就可以直接使用它们，并且语法更加简单，程序代码简洁；同时，MATLAB已经将数学问题的具体算法编成了现成的函数，用户只要熟悉算法的特点、使用场合、函数的调用格式和参数意义等，通过调用函数就可以很快地解决问题。因此，使用MATLAB语言，编程得到了很大简化，编程效率高。

4）MATLAB 使用方便、扩展能力强

MATLAB程序文件是一个纯文本文件，扩展名为.m，用任何字处理软件都可以对它进行编辑。MATLAB本身就像一个解释系统，对其中的函数程序的执行以一种解释执行的方式进行，程序不必经过编译就可以直接运行，而且能够及时报告出现的错误，进行出错原因分析，程序调试容易、编程效率高。由于MATLAB对函数程序的执行是以一种解释的方式进行的，因

而MATLAB完全成了一个开放的系统,用户可以方便地看到函数的源程序,也可以方便地开发自己的程序,甚至创建自己的"库",以扩展MATLAB的功能。

7.2 MATLAB的基本操作

双击系统桌面的MATLAB图标,或者在开始菜单的程序选项中选择MATLAB快捷方式,或者在MATLAB的安装路径的bin子目录中双击可执行文件matlab.exe,都可以启动MATLAB。初次启动MATLAB后,将进入MATLAB默认设置的桌面平台,如图7.1所示(本书主要以MATLAB 6.5版本为例,并具体介绍其有关内容)。



图7.1 MATLAB启动界面

在默认情况下,桌面平台包括5个窗口,分别是MATLAB主窗口、命令窗口、命令历史窗口、当前目录窗口和工作空间窗口。

1. MATLAB主窗口

MATLAB 6.5的主窗口包含其他的4个窗口。主窗口不能进行任何计算任务的操作,只用来进行一些整体的环境参数的设置等。

(1) MATLAB程序路径的设置:单击"File"下拉菜单,点击"Set Path",会弹出图7.2所示的对话窗口,点击"Add Folder",浏览文件夹,找到需要运行程序所在的文件夹位置,按"确定"后窗口中就会新添一条用户设置的路径;按"Save"和"Close",退出对话窗口,以后只要在命令窗口中键入该路径下的MATLAB应用程序,就可以正常运行,不会出现找不到路径的错误了。这是因为MATLAB的一切操作都是在它的搜索路径(包括当前路径)中进行的,如果调用的函数在搜索路径之外,MATLAB则认为此函数并不存在。因此,必须把应用程序所在的目录扩展成MATLAB的搜索路径。

(2) 打开文本编辑器:单击"File"下拉菜单,点击"New",再点击"M-file",会弹出文本编

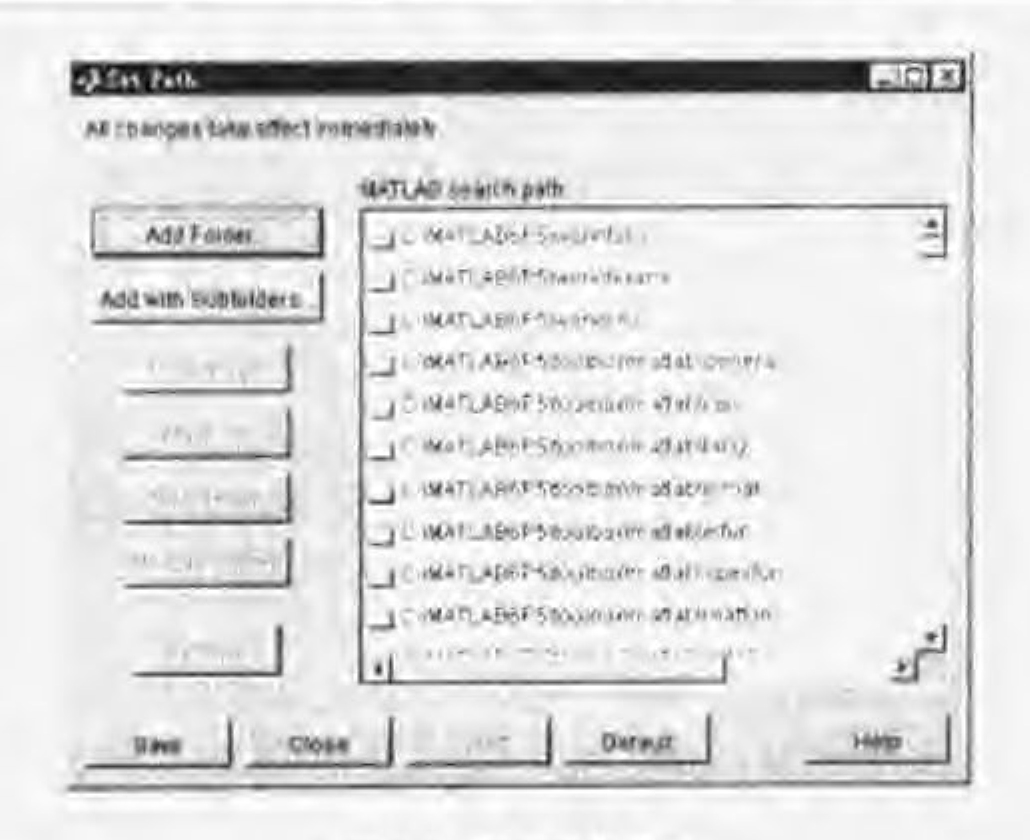


图7-2　程序路径设置

辑窗口,用户可以编写自己的MATLAB应用程序(M 文件和M 函数)。

(3) 清除命令窗口中的内容:单击"Edit"下拉菜单,点击"Clear Command Window",可以清除命令窗口中的所有内容,窗口中只剩下命令提示符">>"。

2. 命令窗口

在MATLAB的命令窗口中,">>"为运算提示符,表示MATLAB正处在准备状态,接受用户的输入指令。当在提示符后输入MATLAB系统命令、MATLAB函数(M函数)、MATLAB应用程序(M文件)和一段MATLAB表达式等,按Enter键后,MATLAB将进行系统管理工作以及进行数值计算、给出计算结果等。如果指令集中调用了MATLAB绘图命令,将会弹出图形窗口,显示计算结果的数学图形。指令完成之后,MATLAB再次进入准备状态。

在MATLAB下进行基本数学运算,只需将运算式直接打入提示号之后,并键入Enter键即可。例如:

```
>> (10 * 19+2/4-34)/2 * 3 (Enter)
ans
=234.7500
```

MATLAB会将运算结果直接存入一变量ans,代表MATLAB运算后的答案,并在屏幕上显示其数值。如果在上例中数学运算式的结尾加上";",则计算结果不会显示在命令窗口上,要得知计算值只需键入表示这一数学运算式的变量名即可。例如:

```
>> a=(10 * 19+2/4-34)/2 * 3; (Enter)
a=
234.7500
```

3. 命令历史窗口

在默认设置下,历史窗口中会保留自安装起所有命令的历史记录,并标明使用时间,这大大方便了使用者的查询。双击某一行命令,即在命令窗口中执行该行命令。

4. 当前目录窗口

在当前目录窗口中可显示或改变当前目录,还可以显示当前目录下的文件并提供搜索功能。在此窗口中,显示并改变当前目录的控件,与主窗口中的路径显示控件完全相同。

5. 工作空间窗口

工作空间窗口是MATLAB的重要组成部分。在工作空间窗口中将显示目前内存中所有的MATLAB变量的变量名、数据结构、字节数以及类型,不同的变量类型分别对应不同的变量名图标。

6. MATLAB 的系统命令

MATLAB系统命令用于对MATLAB系统的管理,最常用的MATLAB系统命令有clear:清理内存变量;clc:清除命令窗口;clf:清除图形窗口;help:在线帮助;helpwin:在线帮助窗口等。

7. 一些常用操作技巧

在MATLAB的使用过程中,通过使用常用的键盘按键(见表7.1)技巧可以使命令窗口的行操作变得简单容易。例如,MATLAB利用了↑↓二个游标键,可以将所用过的指令叫回来重复使用。按下↑则前一次指令重新出现,之后再按Enter键,即再执行前一次的指令。而↓键的功用则是往后执行指令。

另外,Ctrl-C(即同时按Ctrl及C两个键)可以用来中止执行中的MATLAB程序。

表7.1 常用的键盘按键

键盘按键	功　　能	键盘按键	功　　能
↑	Ctrl+p,调用上一行命令	home	Ctrl+a,光标置于当前行开头
↓	Ctrl+n,调用下一行命令	end	Ctrl+e,光标置于当前行末尾
esc	Ctrl+u,清除当前输入行		

8. MATLAB 的标点

在MATLAB语言中,一些标点(见表7.2)被赋予特殊的意义或代表一定的运算。它们被MATLAB变量和语句所应用。

表7.2 常用运算符和特殊字符

标点	定　　义	标点	定　　义
[ ]	方括号,矩阵定义的标志等	:	冒号,具有多种应用功能
,	逗号,区分列及函数参数分隔符等	…	续行符
;	分号,区分行及取消运行显示等	%	百分号,注释标记

9. MATLAB 的帮助系统

MATLAB的帮助系统大致可分为联机帮助系统、命令窗口查询帮助系统和联机演示系

统三大类。下面主要介绍命令窗口查询帮助系统。

help 系列的帮助命令属于命令窗口查询帮助系统，有 help，help＋函数(类)名、helpwin 及 helpdesk，其中后两者是用来调用联机帮助窗口的。

help 命令是最为常用的命令。在命令窗口中直接输入help 命令将会显示当前的帮助系统中所包含的所有项目，即搜索路径中所有的目录名称。

在实际应用中，help＋函数(类)名是最有用的一个帮助命令，可以帮助用户进行深入的学习。例如：

```
>> help log
LOG Natural logarithm.
    LOG(X) is the natural logarithm of the elements of X.
    Complex results are produced if X is not positive.
    See also LOG2, LOG10, EXP, LOGM.
```

7.3 MATLAB 的变量与表达式

1. 数据类型

数据是计算机程序处理的对象。数据可能是整数、实数、复数、数值矩阵或者是字符、字符串等，在MATLAB 语言中，这些数据的类型是相同的。

MATLAB 是一种面向矩阵的编程语言，它将任何数据都看成是矩阵：一个实数是虚部为零的复数，一个复数是1×1 的数值矩阵；向量和字符串等是特殊的矩阵；一个N×M 的数据矩阵由N×M 个复数元素构成。矩阵的类型可以是复数型矩阵、实数型矩阵或由字符组成的字符型矩阵。

在MATLAB 系统中，矩阵有两种不同的存储类型，即全元素矩阵的存储方式或稀疏矩阵的存储方式等。如果MATLAB 变量是全元素矩阵，那么mxArray 阵列结构类型包含有参数pr 和pi。其中，pr 是指向实部向量的指针，pi 是指向虚部向量的指针。对于大小(size)是N×M (N 是矩阵的行(rows)，M 是矩阵的列(columns))的非稀疏复双精度矩阵，矩阵的各数据元素存放在两个双精度(64 位)向量(即内存单元)中：一个存放实部数据(指针 pr 指向它)，另一个存放虚部数据(指针 pi 指向它)。如果是实数，则复双精度矩阵中pi 为空(NULL)。

由此可见，MATLAB 实现变量数据类型的统一，这在很大程度上简化了程序设计。用户不需要事先声明、指定所使用变量的类型、定义变量的维数，MATLAB 会自动根据所赋予变量的值或对变量所进行的操作来确定变量的类型和维数。在赋值过程中，如果变量已存在，则MATLAB 语言将使用新值代替旧值，并以新的变量类型和维数代替旧的变量类型和维数。

2. 变量命名规则

在MATLAB 语言中，变量的命名遵守如下规则：

(1) 变量名以英文字母开头(即第一个字符必须为英文字母)，变量名中可包含字母、数字和下划线“ ”，但不能包含空格符和其他标点符号。

例如，V_31 为合法变量名，而V－31，V 31，V＝31，V＋31，V－31 等都不是合法的变量名。

(2) 变量名中的字母区分大小写。

例如,单一字母X 和x、A 和 a 是不同的变量名,MAT_LAB、MAT_lab 和 mat_lab 等是完全不同的变量名。

(3) 变量名长度不能超过31 个字符(第 31 个字符之后的字符将被忽略)。

例如,V31 为合法变量名。

需要说明的是,MATLAB 语言与其他的程序设计语言一样,也存在变量作用域的问题。在未加特殊说明的情况下,MATLAB 语言将所识别的一切变量视为局部变量,即仅在其调用的函数内有效。若要定义全局变量,应对变量进行声明,即在该变量前加关键字global。一般来说,全局变量习惯用大写的英文字符表示,以便记忆和理解。

3. MATLAB 预定义的变量

MATLAB 有一些预定义的变量,表 7.3 给出了MATLAB 语言中经常使用的一些预定义的变量及其说明。

表 7.3　MATLAB 预先定义的变量

变　量	含　　义	变　量	含　　义
ans	预设的计算结果的变量名	i 或 j	虚数单位 $i=j=\sqrt{-1}$
eps	正的极小值 2.2204e－16	realmax	最大的正实数 1.7977e＋308
pi	内建的 π 值	realmin	最小的正实数 2.2251e－308
inf	∞值,无限大(1/0)	nargin	函数输入参数的个数
NaN	无法定义的一个数(0/0)	nargout	函数输出参数的个数

4. MATLAB 的表达式

MATLAB 数值计算语句是由表达式和变量等组成(即MATLAB 是表达式语言)的,用户输入的语句由MATLAB 系统直接解释运行。因此,变量和表达式是使用MATLAB 进行数值计算的基础。MATLAB 表达式语句有两种最常见的形式:

表达式,或者　变量＝表达式

"表达式"由运算符、函数、变量和数字组成。MATLAB 书写表达式的规则与"手写算式"几乎完全相同。在MATLAB 中,几乎所有的数值计算都必须借助表达式来进行。

在第一种形式中,表达式运算后产生的结果由 MATLAB 系统自动赋给名为ans 的变量,并显示在屏幕上。ans 是一个默认的预定义变量名,它会在以后的类似操作中被自动覆盖掉。所以,对于在后续的计算中将要用到的重要结果,一定要记录下来,应该使用第二种形式的语句(赋值语句)。在第二种形式中,等号右边的表达式计算后产生的结果由MATLAB 系统将其赋给等号左边的变量后放入内存中,并显示在屏幕上。

【例】
```
>> 2*sin(pi/4)+3.^2-sqrt(9)
ans =
    7.4142
>> w=2*sin(pi/4)+3.^2-sqrt(9)
w =
    7.4142
```

注意：

(1) 在书写表达式时，运算符两侧允许有空格，以增加可读性。表达式的末尾可以加上“;”，也可以不加。

(2) 如果一个指令过长可以在结尾加上省略号“…”(代表此行指令与下一行连续)，剩余部分在下一行继续写完。例如：

```
>> S=3*...
6+5
S=    23
```

(3) MATLAB常用算符有：+（加）、-（减）、^（乘幂）、*（乘）、/（右除）、\（左除），在矩阵运算中有左除和右除的区别，对于数字运算则没有区别。

(4) 对于ATLAB语言中的数值显示格式，默认情况下，若数据为整数，则以整型表示；若为实数，则以保留小数点后4位的浮点型表示。在MATLAB系统中，数据的储存和计算都是以双精度进行的，但是用户可以改变数据在屏幕上显示的格式，即可以用format命令控制MATLAB的输出格式，如表7.4所示。应该注意，format命令只影响数据在屏幕上的显示结果，而不会影响数据在MATLAB内的存储和运算。

表7.4　数据输出格式

数据格式命令	含　　义
format/format short	5位定点表示。例如：1.4142
format long	15位定点表示。例如：1.41421356237310
format short e	5位浮点表示。例如：1.4142e+000
format long e	15位浮点表示。例如：1.414213562373095e+000
format short g	系统选择5位定点和5位浮点中更好的表示
format long g	系统选择15位定点和15位浮点中更好的表示
format rational	近似的有理数的表示。例如：sqrt(2)=1393/985
format hex	十六进制的表示。例如：3ff6a09e667f3bcd
format +(plus)	表示大矩阵是分别用+、-和空格表示矩阵中的正数、负数和零
format bank	用元、角、分(美制)定点表示。例如：1.41
format compact	变量之间没有空行
format loose	变量之间有空行

5. 常用数学函数

在MATLAB中，常用的数学函数包括三角函数和双曲函数、指数函数、复数函数、归整函数和求余函数、矩阵变换函数和其他函数。表7.5仅列出了部分常用数学函数。

表7.5　部分常用数学函数

名　称	含　　义	名　称	含　　义	名　称	含　　义
exp	e为底的指数	sqrt	平方根	abs	绝对值
log	自然对数	log10	10为底的对数	log2	2为底的对数

7.4 MATLAB矩阵及运算

7.4.1 矩阵的创建

当需要输入的矩阵维数比较小时，可以直接输入数据建立矩阵。矩阵数据(或矩阵元素)的输入格式如下：

(1) 输入矩阵时要以"[ ]"作为首尾符号，矩阵的数据应放在"[ ]"内部，此时MATLAB才能将其识别为矩阵；

(2) 要逐行输入矩阵的数据，同行数据之间可由空格或","分隔，行与行之间可用";"或回车符分隔；

(3) 矩阵数据可为运算表达式；

(4) 矩阵大小可不预先定义；

(5) 如果不想显示输入的矩阵(作为中间结果)，可以在矩阵输入完成后以";"结束；

(6) 无任何元素的空矩阵也合法。

【例】 a=[1 2 5] 和 a=[1,2,5] 为同一矩阵；

```
b=[3;2;5]  和  b=[3
                  2
                  5] 为同一矩阵。
```

此外，MATLAB提供了一个矩阵编辑器，用户可以用来创建和修改比较大的矩阵。

MATLAB提供了一些生成矩阵的函数，用户可以方便地用他们建立自己所需要的矩阵。

1) 向量、行矩阵、列矩阵的自动生成

用"起始值:增量值:终止值"的格式自动生成等差数列。

【例】
```
>> x=(1:1:10)        % 表示"起始值:增量值:终止值",增量为1时可表示成
                     % "起始值:终止值",即 x=(1:10) 或 x=1:10
x=
    1    2    3    4    5    6    7    8    9    10
>> l=5:15
l=
    5    6    7    8    9    10    11    12    13    14    15
```

用"linspace(起始值:终止值:元素数目)"的格式自动生成等差数列；用"logspace(起始值:终止值:元素数目)"的格式自动生成对数等分数列。

【例】
```
>> y=linspace(30,50,11)
y=
    30    32    34    36    38    40    42    44    46    48    50
```

列矩阵的生成格式如下：

【例】
```
>> x=(1:1:10)';
>> y=linspace(90,95,6)';
```

2) 其他矩阵的自动生成

MATLAB提供了许多特殊矩阵的生成函数，如零矩阵zeros(m,n)，全部元素矩阵ones

(m,n),单位矩阵eye(n),随机矩阵rand(m,n)和魔方矩阵magic(n)等,利用这些矩阵可以生成所需要的矩阵。

【例】 特殊矩阵的生成。

```
>>a=[]                 % 定义空矩阵,即0×0矩阵
a=
                          []
>>zeros(2,2);          % 定义全为0的矩阵(2×2的阵列)
>>ones(3,3);           % 定义全为1的矩阵(3×3的阵列)
```

7.4.2 矩阵的修改

1. 矩阵元素的修改

欲修改矩阵中某个元素的数值,应该先确定该元素的位置,再用赋值语句来实现。

【例】 根据元素在矩阵中的存储顺序来确定矩阵元素的位置,再对元素的数值进行修改。

```
>>x=[1 2 3 4 5 6 7 8;
      4 5 6 7 8 9 10 11];     % 定义2×8矩阵,以分号";"隔离各列元素,该矩阵表示
                              % 了每个向量由2个分量构成的8个向量的多维空间
>> x(3)                       % 找出x的第3个元素,即该矩阵的元素(1,2)
ans =
    2
>> x(1:5)          % 找出x的前5个元素,即第1个到第5个元素
ans =
    1    4    2    5    3
>>x(4)=100         % 给x的第4个元素重新赋值
x=
    1   2   3  4  5  6   7   8
    4  100  6  7  8  9  10  11
>> x(3) = [ ]      % 删除第3个元素,注意矩阵变成了1×15矩阵
x =
    1  4  100  3  6  4  7  5  8  6  9  7
    10  8  11
>> x(16) = 99      % 添加1个元素(加入第16个元素)。
x =
1  4  100  3  6  4  7  5  8  6  9  7
10  8  11  99
```

在MATLAB的内部数据结构中,每一个矩阵都是一个以纵列为主(Column-oriented)的阵列(Array),因此对于矩阵元素的存取,可用一维或二维的索引(Index)来定址。

【例】 根据矩阵行列数来确定矩阵元素的位置,再对元素的数值进行修改。

```
>> z=rand(2,5)
```

```
z=
     0.6299   0.5751   0.0439   0.3127   0.3840
     0.3705   0.4514   0.0272   0.0129   0.6831
>> z(2,3)=0
z=
     0.6299   0.5751   0.0439   0.3127   0.3840
     0.3705   0.4514        0   0.0129   0.6831
```

2. 矩阵行列的修改

A(m,n)表示矩阵第m 行第n 列的元素,中间有","将行和列分开。若要同时表示更多的矩阵元素,可以采用下列表示方法:

A(I,:)	表示矩阵 A 第 I 行全部的元素;
A(:,J)	表示矩阵 A 第 J 列全部的元素;
A([1,3],[2,4])	表示矩阵 A 第 1 行、第 3 行与第 2 列、第 4 列交叉处的元素;
A([1:3],[2:4])	表示矩阵A 第1 行到第3 行与第2 列到第4 列交叉处的元素,也可以写成另外两种形式:A((1:3),(2:4))和 A(1:3,2:4)。

有了矩阵多个元素的表示方法,就可以方便地进行矩阵行和列的修改。

【例】 矩阵行列的修改。

```
>> a=rand(4,5)
a =
     0.0928   0.0158   0.0576   0.6927   0.3533
     0.0353   0.0164   0.3676   0.0841   0.1536
     0.6124   0.1901   0.6315   0.4544   0.6756
     0.6085   0.5869   0.7176   0.4418   0.6992
>> a([1:3],[2:4])=0
a=
     0.0928        0        0        0   0.3533
     0.0353        0        0        0   0.1536
     0.6124        0        0        0   0.6756
     0.6085   0.5869   0.7176   0.4418   0.6992
>> a(4,:)=[ ] % 删除一行元素(第 4 行)
a=
     0.0928   1.0000   0   1.0000   0.3533
     0.0353        0   0        0   0.1536
     0.6124   1.0000   0   1.0000   0.6756
>> a(:,6)=[10;20;30] % 增加一列元素(第 6 列)
a=
     0.0928   1.0000   0   1.0000   0.3533   10.0000
     0.0353        0   0        0   0.1536   20.0000
     0.6124   1.0000   0   1.0000   0.6756   30.0000
```

3. 子矩阵

可以由一个矩阵抽取生成一个子矩阵，或者由几个子矩阵组合生成一个新矩阵。

【例】 矩阵抽取生成子矩阵，子矩阵组合生成新矩阵。

```
>> a=rand(2,3)
a=
    0.8928   0.2548   0.2324
    0.2731   0.8656   0.8049
>> b=ones(2,5);
>> x=[a, b]
x=
    0.8928   0.2548   0.2324   1.0000   1.0000   1.0000   1.0000   1.0000
    0.2731   0.8656   0.8049   1.0000   1.0000   1.0000   1.0000   1.0000
>> c=magic(5);
>> d=c(2,2:4)
d=
    5   7   14
```

7.4.3 MATLAB的矩阵运算

矩阵运算是MATLAB语言最重要和最具有特色的内容之一，是编程进行科学计算的重要基础。矩阵的运算包括矩阵的基本数学运算和矩阵的函数运算。

1. 矩阵的数学运算

矩阵的基本数学运算包括矩阵的算术运算、与常数的运算、转置运算、逆运算、行列式运算、幂运算等。

(1) 矩阵的算术运算：它是指矩阵之间的加、减、乘、除、幂等运算；常数与矩阵的运算，是常数同矩阵的各元素之间进行运算。例如，数加是指矩阵的每个元素都加上此常数，数乘是指矩阵的每个元素都与此常数相乘。需要注意的是，当进行数除时，常数通常只能做除数。

在线性代数中，矩阵没有除法运算，只有逆矩阵。矩阵除法运算是MATLAB从逆矩阵的概念引申而来，主要用于解线性方程组。

例如，设有方程A * X=B，X为变量矩阵，在等式两边同时左乘inv(A)，即：

inv(A) * A * X=inv(A) * B,　　X= inv(A) * B=A\B

即把A的逆矩阵左乘以B，MATLAB就记为A\，称之为“左除”。A、B两矩阵的行数必须相等。如果方程的变量矩阵在左，系数矩阵在右，即X * A=B，则同样有：

X=B * inv(A)=B/A

即把A的逆矩阵右乘以B，MATLAB就记为/A，称之为“右除”。A、B两矩阵的列数必须相等。

在MATLAB中，进行矩阵的幂运算时，矩阵可以作为底数，指数是标量，而矩阵必须是方阵；矩阵也可以作为指数，底数是标量，而矩阵也必须是方阵；但矩阵和指数不能同时为矩阵，否则将显示错误信息。

【例】 矩阵的幂运算。

```
>>a =
    1  2
    3  4
>> a^2          % n 为正整数,a^n 表示矩阵a 自乘n 次
ans =
    7   10
    15  22
>> a^(-2)       % n 为负整数,a^n 表示矩阵a 自乘n 次的逆
ans=
      5.5000  -2.5000
     -3.7500   1.7500
>> 2^a
ans=
    10.4827  14.1519
    21.2278  31.7106
```

(2) 转置运算:在MATLAB 中,矩阵转置运算的表达式和线性代数一样,即对于矩阵A,其转置矩阵的MATLAB 表达式为A′。但应该注意,在MATLAB 中,有几种类似于转置运算的矩阵元素变换运算是线性代数中没有的,即:

fliplr(X) 将X 左右翻转

flipud(X) 将X 上下翻转　rot90(A)

将A 逆时针方向旋转90°

(3) 逆矩阵运算:矩阵的逆运算是矩阵运算中一种重要的运算,在线性代数及计算方法中有很多论述,而在MATLAB 中,求矩阵的逆只需用一个简单的命令inv 就能实现。

(4) 行列式和秩运算:矩阵的行列式的值由det 函数计算得出;矩阵的秩由rank 函数来计算。

【例】 计算矩阵行列式的值和矩阵的秩。

```
>> A =
    1  1  5
    2  2  5
    3  3  5
>> rank(A)
ans =
    2
>> det(A)
ans =
    0
```

2. 矩阵的函数运算

矩阵的函数运算包括基本的数学函数运算、矩阵专门的函数运算以及矩阵的分解运算等内容。

1) 数学函数运算

在MATLAB 中,指数函数expm、对数函数logm、开方函数sqrtm 是把矩阵作为一个整体

进行运算的，即所谓的矩阵运算；其他数学函数都是对矩阵的元素分别进行运算的，即所谓的阵列运算。因此，MATLAB 基本函数库中的大部分常用函数都适合于阵列运算，此时矩阵可以是任意阶。

【例】 >> e=eye(2);

```
>> logm(e)
ans =
    0  0
    0  0
>> e=ones(2);
>> expm(e)
ans =
    4.1945  3.1945
    3.1945  4.1945
>> sqrtm(e)
ans =
    0.7071  0.7071
    0.7071  0.7071
>> a=[0,pi/6,pi/3,pi/2]; b=[-1,0,1];
>> sin(a)
ans =
    0  0.5000  0.8660  1.0000
>> exp(b)
ans =
    0.3679  1.0000  2.7183
```

对其他常用函数的矩阵运算，如三角函数运算和双曲函数运算等，可以采用通用函数形式。在 MATLAB 中使用通用函数的格式为 funm(A,'funname')，其中 A 为输入矩阵变量，funname 为调用的函数名。如funm(b,'log')等同于logm(b)，funm(b,'sqrt')等同于sqrtm(b)，而 funm(b,'sin')，其作用等同于矩阵 b 的正弦运算。

【例】 >> e=ones(2);

```
>>funm(e,'sin')
ans =
    0.4546  0.4546
    0.4546  0.4546
>> sin(e)
ans =
    0.8415  0.8415
    0.8415  0.8415
```

2) 矩阵函数运算

矩阵函数的运算主要包括特征值的计算，奇异值的计算，条件数、各类范数、矩阵迹的计算和矩阵的空间运算等。

【例】 求矩阵的维数和长度。

```
>> a=[10,2,12;34,2,4;98,34,6];
>> size(a) % 求矩阵的维数 (columns & rows)
ans =
    3   3
>> length(a) % 求矩阵的长度,矩阵的长度用向量(或columns)数定义
ans =
    3
```

注意size(a)与length(a)两者之间的区别。

3) 矩阵的分解运算

矩阵分解运算在矩阵运算中具有重要的地位。在用矩阵法求解线性方程组时,就要用到矩阵的分解运算。

(1) 矩阵的LU分解:矩阵LU分解即矩阵的三角分解:A=LU,用于线性方程组的求解。在MATLAB中,LU分解由lu函数实现。

【例】 求A*X=B的解。其中,A=[5,-2,1;1,5,-3;2,1,-5],B=[4;2;-11]。

```
>> [L,U]=lu(A)
L =
    1.0000       0         0
    0.2000    1.0000       0
    0.4000    0.3333    1.0000
U=
    5.0000   -2.0000    1.0000
       0      5.4000   -3.2000
       0        0      -4.3333
>> X=U\(L\B)
X =
    1.0000
    2.0000
    3.0000
```

(2) 矩阵的Chol分解:如果A为n阶对称正定矩阵,则存在一个非奇异下三角实矩阵L,使得$A=LL^T$。当限定L的对角元素为正时,这种分解值是惟一的,称为Chollesky分解。在MATLAB中,这种分解由函数chol实现。

【例】

```
>>a=[4,-1,1;-1,4.25,2.75;1 2.75,3.5];
>> L=chol(a)
L =
    2.0000   -0.5000   0.5000
       0      2.0000   1.5000
       0        0      1.0000
```

(3) 矩阵的QR分解:矩阵的QR分解即矩阵的正交分解。在求解矩阵的特征值时,实阵A可以写成A=QR,其中Q为正交阵,R为上三角阵。若规定R的对角元为正数,则分解惟一。在MATLAB中,QR分解由qr函数实现。

(4) 矩阵的奇异值分解:矩阵的奇异值分解可由函数svd 实现。调用形式为:[U,S,V]=svd(X),生成的U,S 和V 使得X=U ×S×V'。

7.4.4 MATLAB的阵列运算

MATLAB 的运算是以阵列 (array) 运算和矩阵 (matrix) 运算两种方式进行的,而两者在MATLAB 的基本运算性质上有所不同:阵列运算采用的是元素对元素的运算规则,而矩阵运算则是采用线性代数的运算规则。因此,在一些文献中阵列运算被称为数组运算或元素群运算。阵列的运算包括基本运算和函数运算两种形式。

1. 阵列的基本运算

在MATLAB 中,阵列的基本运算采用扩展的算术运算符。即阵列乘.*,阵列除./ 或 .\;阵列幂.^ 。

常数和矩阵之间的数乘运算,即为矩阵元素分别与此常数进行相乘,常数在前在后、加不加"."都一样。常数和矩阵之间的除法运算,对矩阵运算而言常数只能做除数;而对阵列运算而言,由于是"元素对元素"的运算,因此没有任何限制(但一定要加"."运算)。

【例】 a =ones(3,4);

```
>> a * 8;          % a * 8、8 * a、a. * 8 和 8. * a 结果一样
>> a/3             % a/3 和 3\a 都是矩阵除法,结果一样,但 3/a 和 a\3 不合法
ans =
    0.3333   0.3333   0.3333   0.3333
    0.3333   0.3333   0.3333   0.3333
    0.3333   0.3333   0.3333   0.3333
>> 5./a
ans =
    5   5   5   5
    5   5   5   5
    5   5   5   5
>> a./5
ans =
    0.2000   0.2000   0.2000   0.2000
    0.2000   0.2000   0.2000   0.2000
    0.2000   0.2000   0.2000   0.2000
```

阵列的幂运算运算符为".^",它表示每个矩阵元素单独进行幂运算,这是同矩阵的幂运算不同的,矩阵的幂运算和阵列的幂运算所得的结果有很大的差别。

【例】 >> b=[2,2;2,2];

```
>> b^2
ans =
    8   8
    8   8
>> b.^2
ans =
```

```
4   4
4   4
```

2. 阵列的函数运算

在MATLAB中,矩阵按照阵列运算规则进行指数运算、对数运算和开方运算的命令分别是exp,log和sqrt。矩阵进行其他数学函数运算(如三角函数)时,都是按阵列运算规则进行的,矩阵可以是任意阶,其命令通用形式为funname(A),其中funname为常用数学函数名。

【例】 a =ones(3,4);b=zeros(3,4);

```
>> exp(a)
ans =
    2.7183   2.7183   2.7183   2.7183
    2.7183   2.7183   2.7183   2.7183
    2.7183   2.7183   2.7183   2.7183
>> cos(b)
ans=
    1   1   1   1
    1   1   1   1
    1   1   1   1
```

表7.6以具体的计算结果给出了矩阵运算和阵列运算之间的区别。

表7.6 矩阵运算和阵列运算的对比(两个1×3矩阵x和y之间的运算)

矩阵运算		阵列运算	
x	1 2 3	y	4 5 6
x′	1 2 3	y′	4 5 6
x+y	5 7 9	x−y	−3 −3 −3
x+2	3 4 5	x−2	−1 0 1
x * y	Error	x. * y	4 10 18
x′ * y	32	x′. * y	Error
x * y′	4 5 6 8 10 12 12 15 18	x. * y′	Error
x * 2	2 4 6	x. * 2	2 4 6

续表

矩阵运算		阵列运算	
x\y	16/7	x.\y	4 5/2 2
2\x	1/2 1 3/2	2./x	2 1 2/3
x/y	0 0 1/6 0 0 1/3 0 0 1/2	x./y	1/4 2/5 1/2
x/2	1/2 1 3/2	x./2	1/2 1 3/2
x ^ y	Error	x. ^ y	1 32 729
x ^ 2	Error	x. ^ 2	1 4 9
2 ^ x	Error	2. ^ x	2 4 8
(x+i*y)'	1−4i 2−5i 3−6i	(x+i*y).'	1+4i 2+5i 3+6i

7.5 MATLAB字符串

在MATLAB中，所有字符串都要用单引号' '将其界定；变量可以用字符串来赋值，就像变量用数值赋值一样。但要注意，在MATLAB中，字符串中的每个字符(包括空格)都是矩阵的一个元素，字符是以ASCII码储存的。

【例】 >> s='student' % 也可以用s=['student']

s=

student

字符串也可以象数值矩阵一样来连接，形成一个更大的字符串。

【例】 s='student';s1='who';

s=[s,s1] % 连接s与s1

s =

studentwho

【例】 >> str_num='abcde 012345+6789';

>> length(str_num) % 字符串的字符总数

```
ans =
    17
>> size(str_num)   % 把字符串当成一个矩阵
ans =
    1  17
```

(1) 字符串的运算:字符串的运算由字符串函数来实现,有转换运算和操作运算等。部分常用的字符串函数如表 7.7 所示。

表 7.7 部分常用字符串转换表

函数名称	功　　能
fprintf	把格式化的文本写到文件中或显示屏上
num2str	数字转换成字符串
setstr	ASCII 转换成字符串
str2num	字符串转换成数字
eval(string)	作为一个 MATLAB 命令求字符串的值
feval	求由字符串给定的函数值

(2) 字符串和数值之间的转换:函数 num2str 用来把(阿拉伯)数值转换成字符串(数值),再通过字符串连接把所转换的数嵌入到一个字符串句子中。函数 str2num 用来把字符串(数值)转换成(阿拉伯)数值。

【例】
```
>> i=1:4;
>> y=num2str(i)
y =
1 2 3 4        % 此时 y 不再是数字,而是字符,不能用于数值计算
>> 1*y         % y 用 ASCII 值参与计算。
ans =
    49   32   32   50   32   32   51   32   32   52
>> size(y)
ans =
    1   13    % 1×13 字符串矩阵
>> z=str2num(y)
z =
    1   2   3   4   5      % 此时 z 是数字,能用于数值计算
>> 2*z
ans =
    2   4   6   8   10
>> size(z)
ans =
    1   5
```

(3) input 函数:函数 input 能输入一个字符串:x=input(' Enter anything > ',' s ')。这

里，在函数input 里的附加参量' s '告诉MATLAB，“这是一个字符串”。

(4) feval 函数：函数feval 与eval 类似，但在用法上有更多的限制。feval(' fun ',x)求由字符串' fun '给定的函数值，其输入参量是变量x。这说明，函数feval(' fun ',x)等价于求fun(x)的值。函数eval，feval 的基本用途仅限于在用户创建的函数内。一般地，feval 可求出有大量输入参量的函数值，例如，feval(' fun ', x, y, z) 等价于求fun(x, y, z)值。

7.6 MATLAB语句

MATLAB 语句有表达式语句、输入输出语句、控制语句、绘图语句和显示语句等等。这些MATLAB 语句是编程的基础。MATLAB 表达式语句已经介绍，这里只介绍其他的几种语句。

7.6.1 控制语句

(1) for-end 循环语句的一般格式是：

```
for  循环变量= 循环参数表达式
     运算式
end
```

for-end 循环语句的功能是，循环允许一组命令以固定的和预定的次数重复。

循环参数表达式通常是“标量(循环开始参数)：标量(循环终止参数)”或者“标量(循环开始参数)：标量(递增或递减参数)：标量(循环终止参数)”的形式。

【例】

```
n=0:1:10;
y=n;
for i=1:11
    y(i)=sin(n(i));
end
```

具体操作过程是：在for 和end 语句之间的运算式按数组中的每一列(column)执行一次。在每一次迭代中，y 被指定为数组的下一列，即在第n 次循环中，y=array(:, n)。

for-end 循环语句不能通过在循环语句内给循环变量重新赋值来终止循环过程，应该利用break 命令跳出for-end 循环。for-end 循环可按需要嵌套。

为了得到最大的速度，在for-end 循环(while 循环)被执行之前，应预先分配数组。建议最好先使用zeros 或ones 等命令来预先配置所需的内存(即矩阵)大小。

(2) while-end 循环语句的一般格式是：

```
while  条件表达式
     运算式
end
```

while-end 循环语句的功能是，仅仅知道循环产生的条件、而循环次数为不确定的循环运算。

循环条件表达式通常的形式是：

expression rop expression

这里 rop 是 ==, <, >, <=, >= 或 ~=。

可以利用break 命令跳出while-end 循环，while-end 循环可以按需要嵌套。

(3) if-else-end 分支语句的格式之一是：

```
if  条件表达式;
    运算式;
end
```

该 if-else-end 分支语句的功能是，如果在条件表达式中的所有元素为非零，那么就执行 if 和 end 语言之间的语句。

if-else-end 分支语句的格式之二(当有两种选择时)是：

```
if  条件表达式
    运算式1
else
    运算式2
end
```

该 if-else-end 分支语句的功能是，如果条件表达式为真，则执行第一组命令；如果条件表达式是假，则执行第二组命令。

if-else-end 分支语句的格式之三(当有多种选择时)是：

```
if  条件表达式1
        运算式1
elseif  条件表达式2
        运算式2
else  条件表达式3
        运算式3
end
```

该 if-else-end 分支语句的功能是，最后的这种形式，首先检测第一个条件表达式，当条件表达式 1 为真，则执行运算式 1；否则，检测条件表达式 2，依次类推。

(4) switch-case-end 语句的格式是：

```
switch num
case n1
    command_1
case n2
    command_2
case n3
    command_3
    ⋮
otherwise
    command_n
end
```

switch-case-end 语句的的功能是：一旦参数“num”为其中的某个值或字符串时(如 n1 或 n2 或 n3，等等)，就执行所对应的指令(如 commnad_1 或 command_2 或 command_3，等等)；没有对应时，则执行 otherwise 后的语句(command_n)。

7.6.2 输入语句

输入语句比较简单，下面主要通过例子说明输入语句的格式和用法。

输入数值：

```
x=input('please input a number:')
please input a number:22 (Enter)
x=22
```

输入字符串：

```
x=input('please input a string:','s')
please input a string:this is a string (Enter)
x = this is a string
```

MATLAB执行input语句时，会等待用户输入数据，在用户完成数据输入后，然后再执行下一语句。因此，在input括号内的' '中，一定要有提示性的语句，以便MATLAB在屏幕上显示该语句，以提示用户输入数据。如果用户输入字符串数据，还要用加上's'属性。

7.6.3 输出语句

输出显示命令有两种格式：

自由格式 (disp)：

```
disp(23+454-29*4)
361
disp('this is a string')
this is a string
```

格式化输出 (fprintf)：

```
fprintf('The area is %8.5f\n', area)
                        % 注意输出格式前必须要有"%"符号;跳行符号为"\"。
The area is 12.56637    % 这是变量area的格式输出值,含5位小数的8位数。
```

表7.8详细给出了各种格式化输出数据格式的具体含义。

表7.8 格式化输出数据格式(format)

符　号	含　义
%c	Sequence of characters; number specified by field width
%d	Decimal numbers
%e, %f, %g	Floating-point numbers
%i	Signed integer
%o	Signed octal integer
%s	A series of non-white-space characters
%u	Signed decimal integer
%x	Signed hexadecimal integer
[...]	Sequence of characters (scanlist)

7.6.4 辅助语句

MATLAB的辅助语句主要有注释语句、中断语句、暂停语句等。

(1) 注释语句的一般格式是：

% 注释文字

注释语句的功能是，对程序中的语句做必要的说明。注释语句紧跟在被说明语句之后；文字应尽可能中肯、简单、扼要，避免与其他注释相矛盾。

(2) 中断语句的一般格式是：

break

中断语句的功能是，终止一个循环语句的执行过程，即利用break命令跳出for，while循环。

(3) 暂停语句的一般格式是：

pause 或

pause(n)

暂停语句的功能是，pause是程序暂时停止运行，直到按下回车键，继续执行程序；而pause(n)是中断n秒后，程序自动继续执行。

请注意，Ctrl-C键(即同时按Ctrl及C两个键)是用来中止执行中的MATLAB的工作。

7.6.5 回显语句

回显语句的一般格式是：

echo on/off

回显语句的功能是，控制是否在屏幕上回显MATLAB的正在执行的语句。系统默认的状态是echo off。该语句对于调试程序很有帮助。

7.6.6 绘图语句

在MATLAB中，使用绘图语句绘图非常简单，在后面专门介绍。

7.7 M文件与M函数

在MATLAB命令窗口中，键入一行命令，系统会立刻执行该命令。这种人机交互的工作方式称为命令行运行模式。当运行的命令较多时，如果采用命令行运行模式，直接从键盘上逐行输入命令显然比较麻烦，并且程序可读性差、难以存储，此时应该采用M程序运行模式。所谓M程序运行模式，是指由MATLAB语句构成程序、以ASCII码文本文件的形式存储、用.m作为文件扩展名的MATLAB程序在命令窗口中的自动运行。MATLAB程序可分成M文件和M函数两种：M文件即命令文件(script file)，它是用户为解决问题自己编制的程序；M函数即函数文件(function file)，它是一种子程序，一般由其他程序调用。

7.7.1 M文件

MATLAB向用户提供了一个自主编写程序的环境，用户可以根据自己的需要，灵活运用MATLAB的函数(M函数)或者命令编程。例如，输入如下一段程序：

```
t=-10:0.1:10;
wc=10;
ft=1/2*sin(t);
ft1=ft.*cos(wc*t);d=0:length(ft)-1;
plot(d,ft1); % 绘图语句
```

写完文件用db.m文件名保存(save)后,在命令窗口中键入文件名db,则显示出运行该文件的结果。这里要注意文件db.m的路径是否设置正确。

编写M文件的一般格式是:用clear、close all等语句开头,清除掉工作空间中原有的变量和图形,以免其他已执行过的程序残留数据对本程序的影响;文件名长度一般不要超过8个字符(英文字母、数字和下划线),文件扩展名要用.m。

7.7.2 M函数

M函数是MATLAB程序的一种形式,可以以函数调用的方式调用。它和M文件之间的差别是,它由function开头,后跟的函数名与文件名相同;有输入/输出变量,可进行变量传递;除非用global声明,程序中的变量均为局部变量,不保存在工作空间中。以下是一个M函数示例:

```
function [I,h]=Trapezd(f_name,a,b,n)
%  M函数的定义语句格式:等式左边的中括号内的参数是全局形式参数
%                      等式右边的小括号内的参数是局部变量(实参)
function [I,h]=Trapezd(f_name,a,b,n)
h=(b-a)/n;
x=a+(0:n)*h;
f=feval(f_name,x);
I=h*(sum(f)-(f(1)+f(length(f)))/2);
```

M函数编写完成后,一般用M函数名作为文件名(如trapezd.m)来保存文件

7.8 数学图形的绘制

在科学研究和工程设计中,数值计算的结果一般是以数据的形式反映客观世界的规律,但人们往往希望这些数据结果能够直观化、可视化,以便更好地理解这些数据之间的规律,MATLAB的图形处理功能就是用数学图形来反映数据之间所存在的客观规律。

7.8.1 二维数学图形绘制

在MATLAB中,最常用、最基本的绘图是二维数学图形的绘制。表7.9列出了二维图形函数库中的基本图形函数。另外,在通用图形函数库(graphics)中,cls,line等也是用得最多的图形函数。

1) plot绘图函数

MATLAB最基本、最重要的绘图命令就是plot绘图函数。它有多种基本的调用格式。

```
plot(Y)   %  如果Y是实数,Y就是它的列(column);如果是复数,则相当于plot(real
              (Y),imag(Y))
plot(X,Y,…)              %  绘制X为横坐标,Y为纵坐标的数学图形
```

```
plot(X,Y,X1,Y1,…)        % 同时绘制Y对X、Y1对X1的数学图形
plot(X,Y,LineSpec,…)     % 绘图不同线型、标识、颜色等的数学图形
```

表 7.9　二维图形函数库(graph2d)

	函数名	函数功能	函数名	函数功能
基本X-Y图形	plot	线性X-Y坐标绘图	polar	极坐标绘图
	loglog	双对数X-Y坐标绘图	plotyy	用左、右两种Y坐标画图
	semilogx	半对数X坐标绘图	semilogy	半对数Y坐标绘图
坐标控制	axis	控制坐标轴比例和外观	subplot	在平铺位置建立图形轴系
	hold	保持当前图形		
图形注释	title	标出图名(适用于三维图形)	gtext	用鼠标定位文字
	xlabel	X轴标注(适用于三维图形)	legend	标注图例
	ylabel	Y轴标注(适用于三维图形)	grid	图上加出标网格(适用于三维图形)
	text	在图上标文字(适用于三维图形)		
打印	print	打印图形或把图存为M文件	orient	设定打印纸方向

【例】

```
t=0:0.001:10;
y=sin(t);
Y=sin(10*t);
c=y.*Y;
plot(t,y,'r:',t,Y,'b')
```

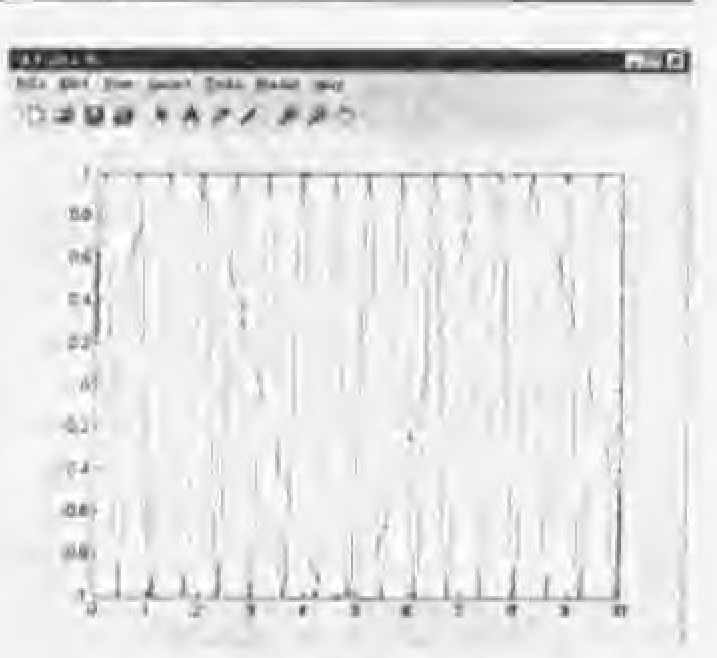


图 7.3　数学函数图形

在本例中,用绘图语句plot函数的不同参数,绘制出线型和颜色不同的两个数学图形,如图7.3所示。两条曲线各自有一对变量,并且有描述颜色的参数r(red)和b(blue)。

2) line 绘图函数

在MATLAB中,绘制直线,使用line命令。例如:

```
line([0,5],[0,10]) % 绘制点(0,0)到点(5,10)的直线。
```

3) hold 和 clf 图形函数

在绘图过程中,如果要在已经绘制的图形上添加新的图形,则可以使用hold命令来实现图形的保持功能。hold on 表示启动图形保持功能,hold off 表示关闭图形保持功能。

在绘图过程中,为了彻底清除前面图形的影响,应该在绘图语句的前面使用clf命令。

7.8.2　数学图形属性修改

1) 颜色和线型修改

表7.10是plot绘图函数的若干参数。若要改变颜色,则在座标对的后面加上相关字符串即可;若要同时改变颜色及线型(Line style),则也是在坐标对的后面加上相关字符串即可。

表7.10 plot绘图命令参数

字 元	颜 色	字 元	图 线 型 态
y	yellow 黄色	.	point
k	black 黑色	o	circle
w	white 白色	x	cross
b	blue 蓝色	+	plus sign
g	green 绿色	*	asterisk
r	red 红色	—	Real line
c	cyan 亮青色	:	dot
m	amethyst 锰紫色	—.	Point-broken line
		——	Broken line

【例】 t=0:0.01:10;plot(t,sin(t),'r') % 效果如图7.4所示。
plot(t,sin(t),'r*') % 效果如图7.5所示。

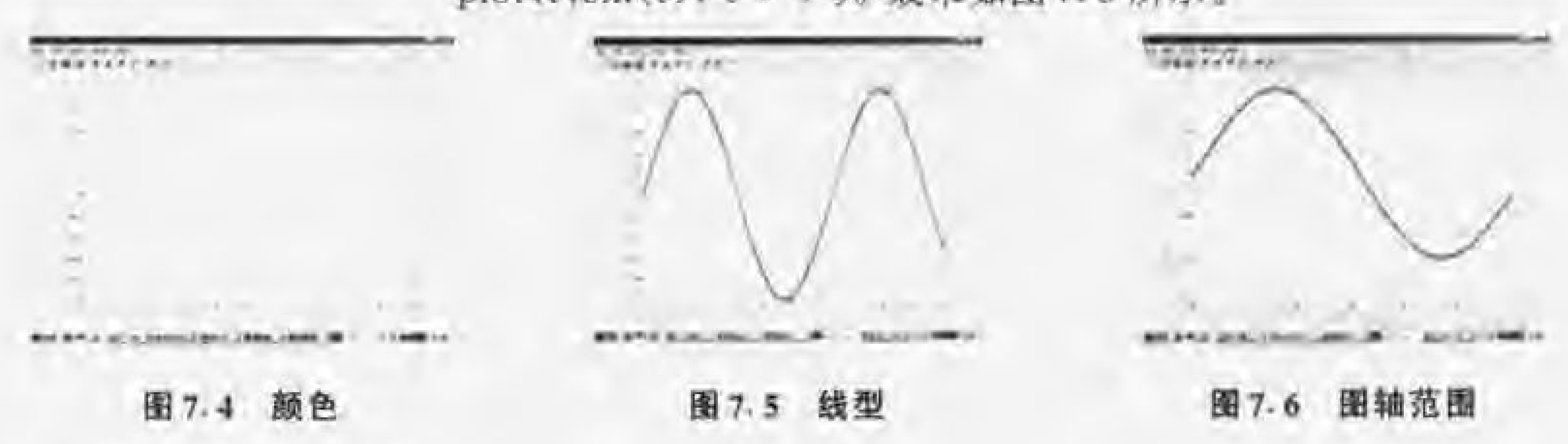


图7.4 颜色　　图7.5 线型　　图7.6 图轴范围

2）调整图轴的范围

用axis([xmin,xmax,ymin,ymax])函数来调整图轴的范围。

【例】 axis([0,6,−1.5,1]) % 效果如图7.6所示。

3）图轴、标题标注与文本标注

x轴、y轴及图形标题标注命令格式为：

```
xlabel('Input Value'); ylabel('Function Value'); title('this is a function');
```

文本标注命令格式为：

```
text(0.2,0.8,'f(x)=sin(x)+cos(2x)')
```

效果如图7.7所示。

4）图例标注与显示格线

图例标注与显示格线的格式：

```
legend('y = sin(x)')   % 指定图形的式样。可以是线图(line plots)，棒图(bar graphs)，
                       % 饼图(pie charts)等
grid on                % 在当前图形上加栅格线
```

7.8.3 绘制矩阵的图形

绘制矩阵的图形，可以用plot命令。在MATLAB中，M文件peaks.m建立了一个矩阵线图的数据，用plot(M文件名)语句即可实现矩阵线图的绘制：

plot(peaks);

屏幕显示如图 7.8 所示。



图 7.7　图轴、标题与文本标注

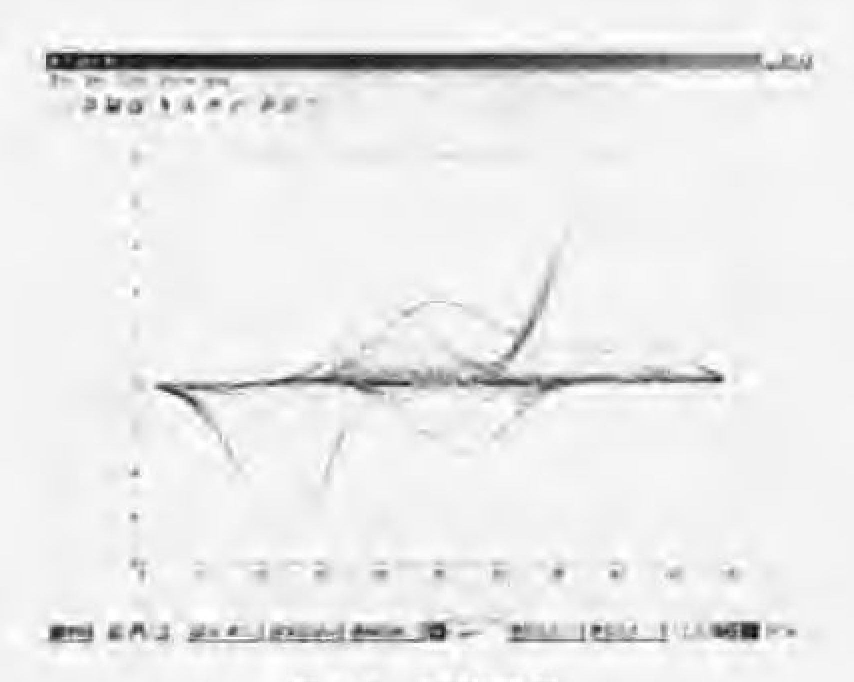


图 7.8　矩阵图形

7.8.4　三维数学图形绘制

1）绘制三维曲线

绘制三维曲线的命令格式为：

plot3(x1,y1,z1,LineSpec,…)

函数格式除了包括第三维的信息(比如 Z 方向)之外，其他与二维函数 plot 相同。其功能是：plot3 语句将绘制二维图形的函数 plot 的特性扩展到三维空间。

【例】　t=(0:0.1:3)*pi;

x=sin(t);

y=cos(t);

z=tan(t);

plot3(x,y,z,'bo—');

效果如图 7.9 所示。

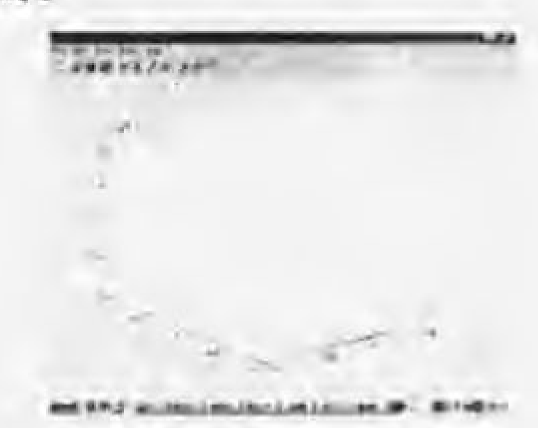


图 7.9　三维曲线

2）绘制三维曲面

绘制三维曲面的命令格式为：

mesh(z)　或

mesh(...,C)　或

meshc(...)　或

meshz(...)　等

mesh(z) 语句按照 x = 1:n 和 y = 1:m 绘制三维图。在这里，[m,n]=size(Z)，Z 是高度，用不同的颜色表示该高度值。



图 7.10　三维曲面

【例】　x=—2:0.01:2;

[x,y]=meshgrid(x,x);

% 为绘制三维图形而从 x 生成的 x

% 和 y 矩阵。[x,y]是 401x401 的矩阵。

r=sqrt(x.^2+x.^2)+eps;

z=sinc(r);

mesh(z);

效果如图7.10所示。

mesh(...,C)语句中,参数C代表颜色的数值。如果x,y,和z是矩阵,那么C必须是一个相同大小的色标矩阵。

meshc(...)语句绘制网格轮廓线图。

【例】 [X,Y] = meshgrid(-3:.125:3);

Z = peaks(X,Y);% 为了方便绘制三维图,MATLAB提供了一个peaks函数,可产生一
% 个凹凸有序的曲面,包含了3个局部极大点及3个局部极小点

meshc(X,Y,Z); axis([-3 3 -3 3 -10 5])

效果如图7.11所示。



图7.11 三维图形 图7.12 三维图形

meshz(...)语句依参考平面绘制网格四周门帘线图(a curtain plot around the mesh)。

【例】 [X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y); meshz(X,Y,Z)

效果如图7.12所示。

【例】 绘制三维曲面之例一,效果如图7.13所示。

```
[x,y,z] =peaks;
subplot(2,2,1);
meshz(x,y,z);                    % 曲面加上门帘线
axis([-inf inf -inf inf -inf inf]);
subplot(2,2,2);
waterfall(x,y,z);                % 在x方向产生水流效果
axis([-inf inf -inf inf -inf inf]);
subplot(2,2,3);
meshc(x,y,z);                    % 同时画出网状图与等高线
axis([-inf inf -inf inf -inf inf]);
subplot(2,2,4);
surfc(x,y,z);                    % 同时画出曲面图与等高线
axis([-inf inf -inf inf -inf inf]);
```

【例】 绘制三维曲面之例二,效果如图7.14所示。

subplot(2,2,1)

```
contour3(peaks,50);               % 画出曲面在三度空间中的等高线
axis([-inf inf -inf inf -inf inf]);
subplot(2,2,2)
contour(peaks, 50);               % 画出曲面等高线在XY平面的投影
subplot(2,2,3)
t=linspace(0,20*pi, 501);
plot3(t.*sin(t), t.*cos(t), t);          % 画出三度空间中的曲线
subplot(2,2,4)
plot3(t.*sin(t), t.*cos(t), t, t.*sin(t), t.*cos(t), -t);
                                         % 同时画出两条三度空间中的曲线
```



图7-13　三维曲面

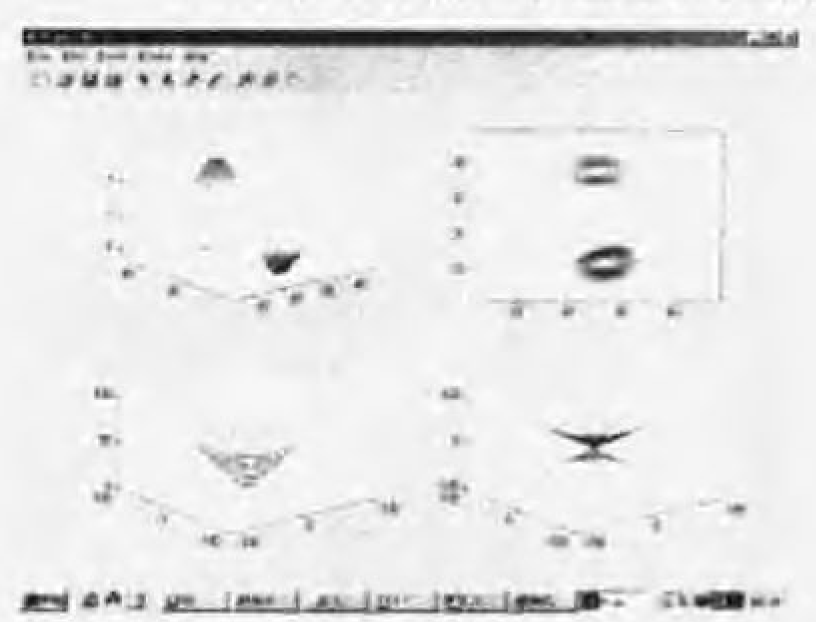


图7-14　三维曲面

附　　录

附录A　常用MATLAB程序

本附录中给出了部分常见的数值计算方法的MATLAB程序，供读者学习与参考。

A.1　非线性方程求根程序

1. 绘制函数图像程序

程序名称　fxg.m

调用格式　fxg('f_name',xmin,xmax,n_points)

程序功能　绘制给定函数y=f(x)的图像。

输入变量　f_name为用户自己编写给定函数y=f(x)的M函数而命名的程序文件名；

xmin为图形横坐标轴的最小值；

xmax为图形横坐标轴的最大值；

n_points为自变量X的采样数。

程序：

```
function fxg(f_name,xmin,xmax,n_points)
f_name;
dx=(xmax-xmin)/n_points;
xp=xmin:dx:xmax;
yp=feval(f_name,xp);
plot(xp,yp,'r');      % draw y=f(x)
xlabel('x');ylabel('f(x)');title('y=f(x)');hold on;
yp=0.0*xp;
plot(xp,yp)           % draw x axis
```

例A.1　绘制函数$f(x)=e^x-5x^2$的图像。

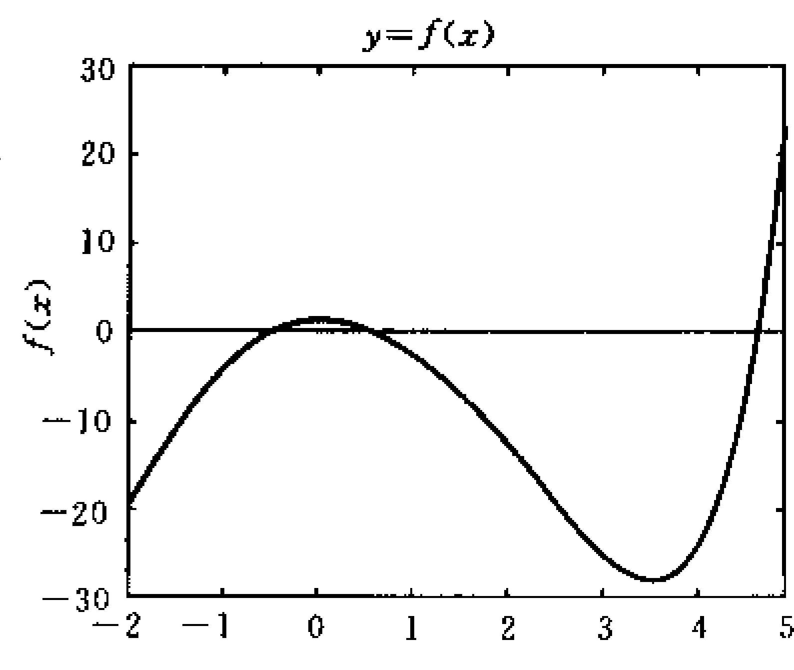
解　编写$f(x)=e^x-5x^2$的M函数，取文件名为ex_5x2.m：

```
function y=ex_5x2(x)
y=exp(x)-5.*x.^2;
```

调用程序fxg.m绘图的格式为：

```
fxg('ex_5x2',-2,5,100)
```

计算结果如图A.1所示。



图A.1　函数$f(x)=e^x-5x^2$的图像

2. 对分法程序

程序名称　bisec_g. m

调用格式　bisec_g('f_name',a,c,xmin,xmax,n_points)

程序功能　用对分法求非线性方程的根,并用示意图表示求根过程。

输入变量　f_name 为用户自己编写给定函数 y=f(x)的 M 函数而命名的程序文件名;

[a,c]为含根区间;

xmin 为图形横坐标轴的最小值;

xmax 为图形横坐标轴的最大值;

n_points 为自变量 X 的采样数。

程序:

```
function bisec_g(f_name,a,c,xmin,xmax,n_points)
 %      a,c : end points of initial interval
 %      tolerance : tolerance
 %      it_limit : limit of iteration number
 %      Y_a,Y_c : y values of the current end points
clf,hold off
clear Y_a,clear Y_c
wid_x=xmax-xmin;dx=(xmax-xmin)/n_points;
xp=xmin:dx:xmax;
yp=feval(f_name,xp);
plot(xp,yp,'r');                 % y=f(x)
xlabel('x');ylabel('f(x)');title('Bisection Method'),hold on
ymin=min(yp);ymax=max(yp);wid_y=ymax-ymin;
yp=0.0*xp;
plot(xp,yp)                      % x axes
fprintf('Bisection Scheme \n\n') ;
fprintf('It     a        b        c        fa=f(a)');
fprintf('  fc=f(c)  abs(fc-fa)  abs(c-a)/2\n');
tolerance=0.000001;it_limit=30;
it=0;
Y_a=feval(f_name,a);Y_c=feval(f_name,c);
plot([a,a],[Y_a,0],'black');text(a,-0.1*wid_y,'x=a')   %  line
plot([c,c],[Y_c,0],'black');text(c,-0.3*wid_y,'x=c')   %  line
if(Y_a*Y_c>0)
   fprintf('f(a)f(c)>0\n');
else
while 1
        it=it+1;
        b=(a+c)/2;
```

```
            Y_b=feval(f_name,b);
            plot([b,b],[Y_b,0],':');      %  line
            plot(b,0,'o')                 %  o point
        if it<4
            text(b,wid_y/20,[num2str(it)])
        end
            fprintf('%3.0f %10.6f %10.6f',it,a,b);
            fprintf('%10.6f %10.6f %10.6f',c,Y_a,Y_c);
            fprintf(' %12.3e %12.3e\n',abs(Y_c-Y_a),abs(c-a)/2);
        if(abs(c-a)/2<=tolerance)
            fprintf('Tolerance is satisfied. \n');
            break
        end
        if(it>it_limit)
            fprintf('Iteration limit exceeded. \n');
            break
        end
        if (Y_a * Y_b<=0)
            c=b;Y_c=Y_b;
        else
            a=b;Y_a=Y_b;
        end
    end
    fprintf ('Final result ; Root = %12.6f \n',b)
    end
    x=b;
    plot ([x,x],[0.05 * wid_y,0.2 * wid_y],'g'); text (x,0.25 * wid_y,'Final solution')
    plot([x,(x-wid_x * 0.004)],[0.05 * wid_y,0.09 * wid_y],'r') % arrow line
    plot([x,(x+wid_x * 0.004)],[0.05 * wid_y,0.09 * wid_y],'r') % arrow line
```

例A.2　求$f(x)=e^x-5x^2=0$在区间[0,1]上的根。

解　编写$f(x)=e^x-5x^2$的M函数,取文件名为ex_5x2.m:

```
function y=ex_5x2(x)
y=exp(x)-5.*x.^2;
```

调用程序bisec_g.m求根的格式为:

```
bisec_g('ex_5x2',0,1,0,1,200)
```

求根过程如图A.2所示,计算结果如下:

Bisection Scheme

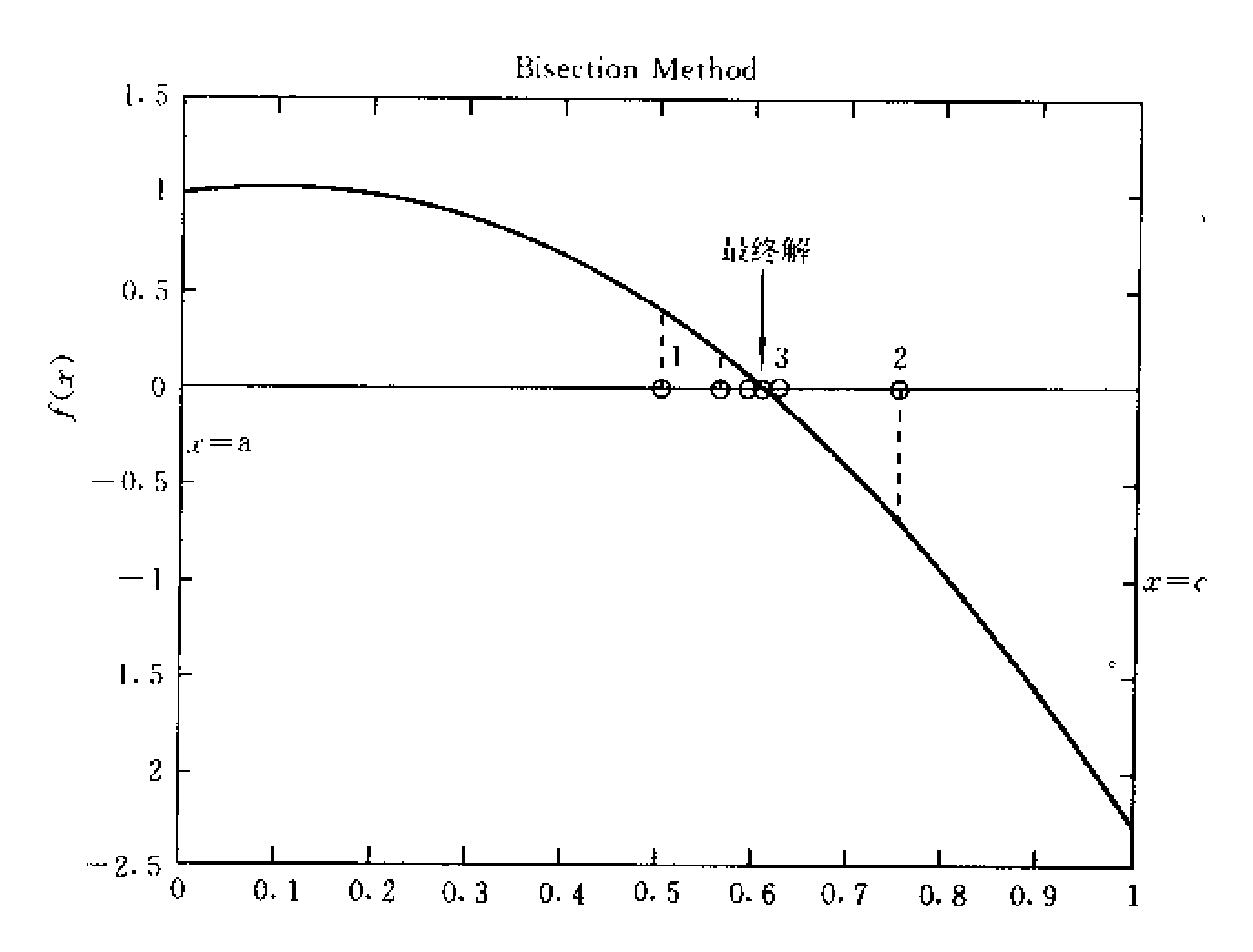


图 A.2　对分过程示意图

It	a	b	c	fa=f(a)	fc=f(c)	abs(fc−fa)	abs(c−a)/2
1	0.000000	0.500000	1.000000	1.000000	−2.281718	3.282e+000	5.000e−001
2	0.500000	0.750000	1.000000	0.398721	−2.281718	2.680e+000	2.500e−001
3	0.500000	0.625000	0.750000	0.398721	−0.695500	1.094e+000	1.250e−001
4	0.500000	0.562500	0.625000	0.398721	−0.084879	4.836e−001	6.250e−002
5	0.562500	0.593750	0.625000	0.173023	−0.084879	2.579e−001	3.125e−002
6	0.593750	0.609375	0.625000	0.048071	−0.084879	1.329e−001	1.563e−002
7	0.593750	0.601563	0.609375	0.048071	−0.017408	6.548e−002	7.813e−003
8	0.601563	0.605469	0.609375	0.015581	−0.017408	3.299e−002	3.906e−003
9	0.601563	0.603516	0.605469	0.015581	−0.000851	1.643e−002	1.953e−003
10	0.603516	0.604492	0.605469	0.007380	−0.000851	8.232e−003	9.766e−004
11	0.604492	0.604980	0.605469	0.003268	−0.000851	4.120e−003	4.883e−004
12	0.604980	0.605225	0.605469	0.001210	−0.000851	2.061e−003	2.441e−004
13	0.605225	0.605347	0.605469	0.000179	−0.000851	1.031e−003	1.221e−004
14	0.605225	0.605286	0.605347	0.000179	−0.000336	5.153e−004	6.104e−005
15	0.605225	0.605255	0.605286	0.000179	−0.000078	2.576e−004	3.052e−005
16	0.605255	0.605270	0.605286	0.000051	−0.000078	1.288e−004	1.526e−005
17	0.605255	0.605263	0.605270	0.000051	−0.000014	6.441e−005	7.629e−006
18	0.605263	0.605267	0.605270	0.000018	−0.000014	3.220e−005	3.815e−006
19	0.605267	0.605268	0.605270	0.000002	−0.000014	1.610e−005	1.907e−006
20	0.605267	0.605268	0.605268	0.000002	−0.000006	8.051e−006	9.537e−007

Tolerance is satisfied.

Final result ; Root = 0.605268

3. Newton 迭代法程序

程序名称　Newt_g.m

调用格式　x=Newt_g('f_name',x0,xmin,xmax,n_points)

程序功能　用Newton迭代法求非线性方程的根，并用示意图表示迭代过程。

输入变量　f_name为用户自己编写给定函数y=f(x)的M函数而命名的程序文件名；
　　x0为根的近似迭代初值；
　　xmin为图形横坐标轴的最小值；
　　xmax为图形横坐标轴的最大值；
　　n_points为自变量X的采样数。

输入变量　x为方程的根。

程序：

```
function x=Newt_g(f_name,x0,xmin,xmax,n_points)
clf,hold off
wid_x=xmax-xmin;dx=(xmax-xmin)/n_points;
xp=xmin:dx:xmax;
yp=feval(f_name,xp);
plot(xp,yp,'r');              %  y=f(x)
xlabel('x');ylabel('f(x)');title('Newton Iteration'),hold on
ymin=min(yp);ymax=max(yp);wid_y=ymax-ymin;
yp=0.0*xp;
plot(xp,yp)                   %  x axes
x=x0;
xb=x+999;
n=0;
del_x=0.001;
while abs(x-xb)>0.000001
if n>300
      break;
end
     y=feval(f_name,x);
     plot([x,x],[y,0],'black');
     plot(x,0,'o')
     fprintf('n=%3.0f,    x=%12.5e,    y=%12.5e\n',n,x,y);
if n<4
     text(x,-wid_y/10,[num2str(n)]),
end
     y_driv=(feval(f_name,x+del_x)-y)/del_x;  %  derivative
     xb=x;
     x=xb-y/y_driv;
     n=n+1;
     plot([xb,x],[y,0],'g')                   %  tangent
end
```

```
plot ([x, x], [0. 05 * wid_y, 0. 2 * wid_y],'r'); text (x, 0. 25 * wid_y,'Final solution')
plot([x,(x-wid_x * 0. 004)],[0. 01 * wid_y,0. 09 * wid_y],'r')
plot([x,(x+wid_x * 0. 004)],[0. 01 * wid_y,0. 09 * wid_y],'r')
```

例 A. 3 求$f(x)=e^x-5x^2=0$在区间[3,5]上的根。

解 编写$f(x)=e^x-5x^2$的M函数,取文件名为ex_5x2. m:

```
function y=ex_5x2(x)
y=exp(x)-5. * x. ^ 2;
```

调用程序Newt_g. m求根的格式为:

```
x=Newt_g('ex_5x2',3. 7,3,5,100)
```

求根过程如图A. 3所示,计算结果如下:

```
n=   0,   x=4. 50000e+000,   y=-1. 12329e+001
n=   1,   x=4. 74930e+000,   y=2. 72433e+000
n=   2,   x=4. 70928e+000,   y=8. 54043e-002
n=   3,   x=4. 70794e+000,   y=1. 57526e-004
n=   4,   x=4. 70794e+000,   y=1. 24833e-007
x=4. 7079
```

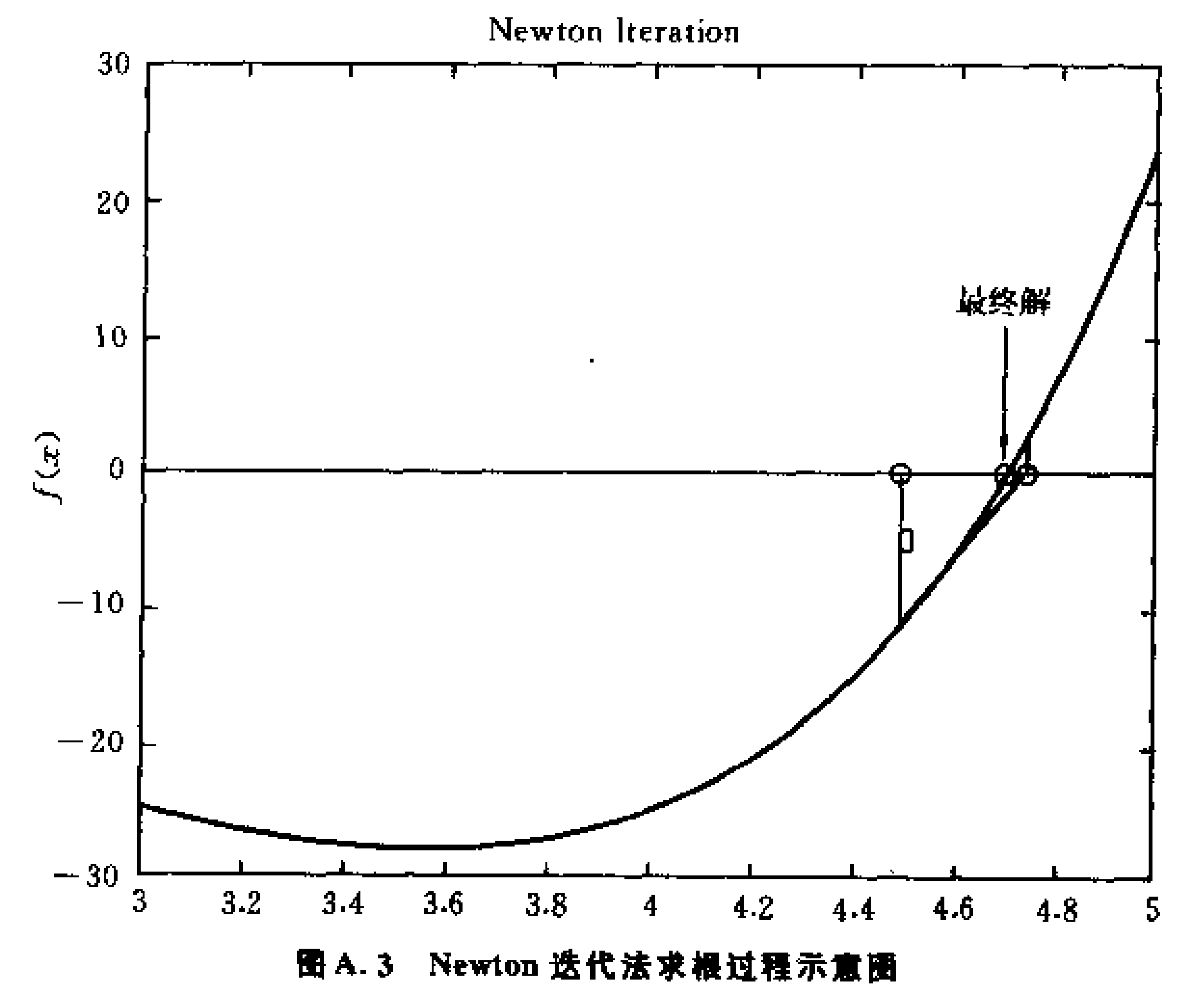


图 A. 3 Newton 迭代法求根过程示意图

A. 2 插值方法与曲线拟合方法程序

1. Lagrange 插值法程序

程序名称 lagrange. m

调用格式 y=lagrange(x0,y0,x)

程序功能 求插值点x处的函数值y。

输入变量　x0,y0 为已知的离散数据；

　　　　　x 为插值节点。

输入变量　y 为插值节点对应的函数值。

程序：

```
function y=lagrange(x0,y0,x)
n=length(x0);m=length(x);
for i=1:m
z=x(i);
s=0.0;
for k=1:n
  p=1.0;
    for j=1:n
    if j~=k
    p=p*(z-x0(j))/(x0(k)-x0(j));
    end
    end
  s=p*y0(k)+s;
end
y(i)=s;
end
```

例 A.4　已知x=[0.4: 0.1: 0.8];

y=[-0.916291 -0.693147 -0.510826 -0.356675 -0.223144];

求插值点 0.54 对应的函数值。

解　lagrange(x,y,0.54)

ans = -0.6161

2. 分段线性插值函数和曲线拟合函数

实现分段线性插值不需编制函数程序,MATLAB 自身提供了内部函数interpl。调用格式如下：

yi=interpl(x,y,xi)

对一组节点(x,y)进行插值,计算插值点xi 的函数值。x 为节点向量值,y 为对应的节点函数值。如果y 为矩阵,则插值对y 的每一列进行;若y 的维数超出 x 或Xi 的维数,则返回NaN。

实现曲线拟合亦不需编制函数程序,MATLAB 自身提供了内部函数polyfit。调用格式如下：

c= polyfit (x,y,n)

对一组节点(x,y)进行拟合,计算拟合系数此c,x 为节点向量值,y 为对应的节点函数值。

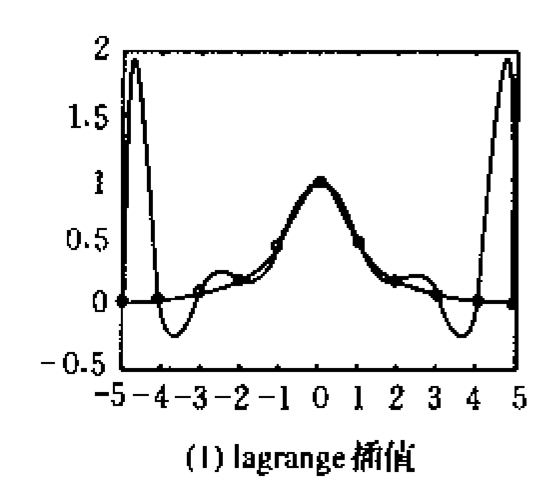
例 A.5　用lagrange 插值程序,分段线性插值函数和曲线拟合函数分别研究 y=1./(1+x.^2)的插值、分段插值和拟合问题。

解　程序文件命名为Runge.m,编制MATLAB 程序如下：

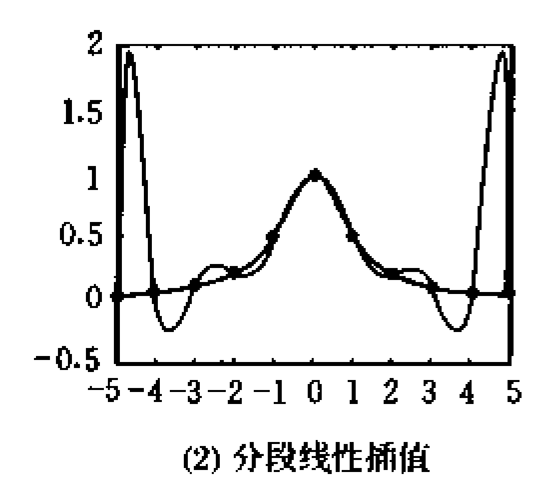
```
clear
```

```
x=linspace(-5,5,11)
y=1./(1+x.^2)
xn=[-5:0.05:5];
yn=lagrange (x,y,xn);
fn=1./(1+xn.^2);
plot (xn,yn,'r')
hold on
plot (xn,fn,'-b')
hold on
plot (x,y,'o')
hold on
i=input('switch i=1,2,3?')
switch i
    case 1
        yp=interp1(x,y,xn);
        plot(xn,yp,'*m')
        hold on
    case 2
        c=polyfit(x,y,8);
        zn=polyval(c,xn);
        plot(xn,zn,'xg')
        hold on
    otherwise
end
```

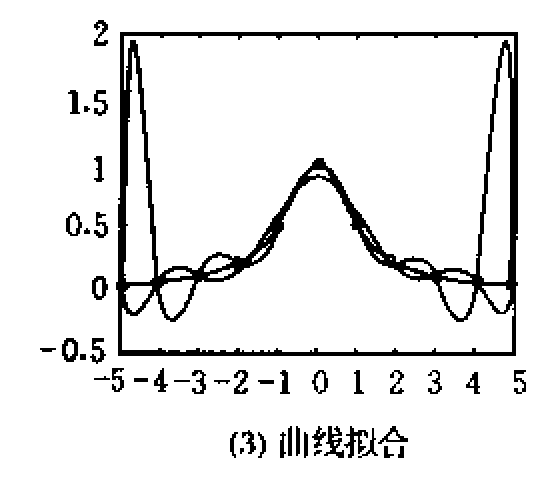
运行 Runge.m 程序,y=1./(1+x.^2)的 lagrange 插值、分段插值和拟合结果如图 A.4 所示。



(1) lagrange插值



(2) 分段线性插值



(3) 曲线拟合

图 A.4 插值方法和拟合方法对比

A.3 数值积分程序

1. 复化梯形积分法程序

程序名称 Trapezd.m

调用格式 I=Trapezd('f_name',a,b,n)

程序功能　用复化梯形公式求定积分值。

输入变量　f_name 为用户自己编写给定函数y=f(x)的M 函数而命名的程序文件名；
　　a 为积分下限；
　　b 为积分上限；
　　n 为积分区间[a,b]划分成小区间的等份数。

输出变量　I 为定积分值。

程序：

```
function I=Trapezd(f_name,a,b,n)
h=(b-a)/n;
x=a+(0:n)*h;
f=feval(f_name,x);
I=h*(sum(f)-(f(1)+f(length(f)))/2);
hc=(b-a)/100;
xc=a+(0:100)*hc;
fc=feval(f_name,xc);
plot(xc,fc,'r');
hold on ;
title('Trapezoidal Rule');xlabel('x');ylabel('y');
plot(x,f);
plot(x,zeros(size(x))) ;
for i=1:n;
    plot([x(i),x(i)],[0,f(i)]);
end
```

例A.6　求$I=\int_0^{\pi}\sin x\mathrm{d}x$。

解　先编制$y=\sin x$的M 函数，程序文件命名为sin_x.m：

```
function y=sin_x(x)
y=sin(x);
```

将区间4等分，调用格式为：

```
I=Trapezd('sin_x',0,pi,4)
```

计算结果为：

```
I=1.8961
```

将区间20等分，调用格式为：

```
I=Trapezd('sin_x',0,pi,20)
```

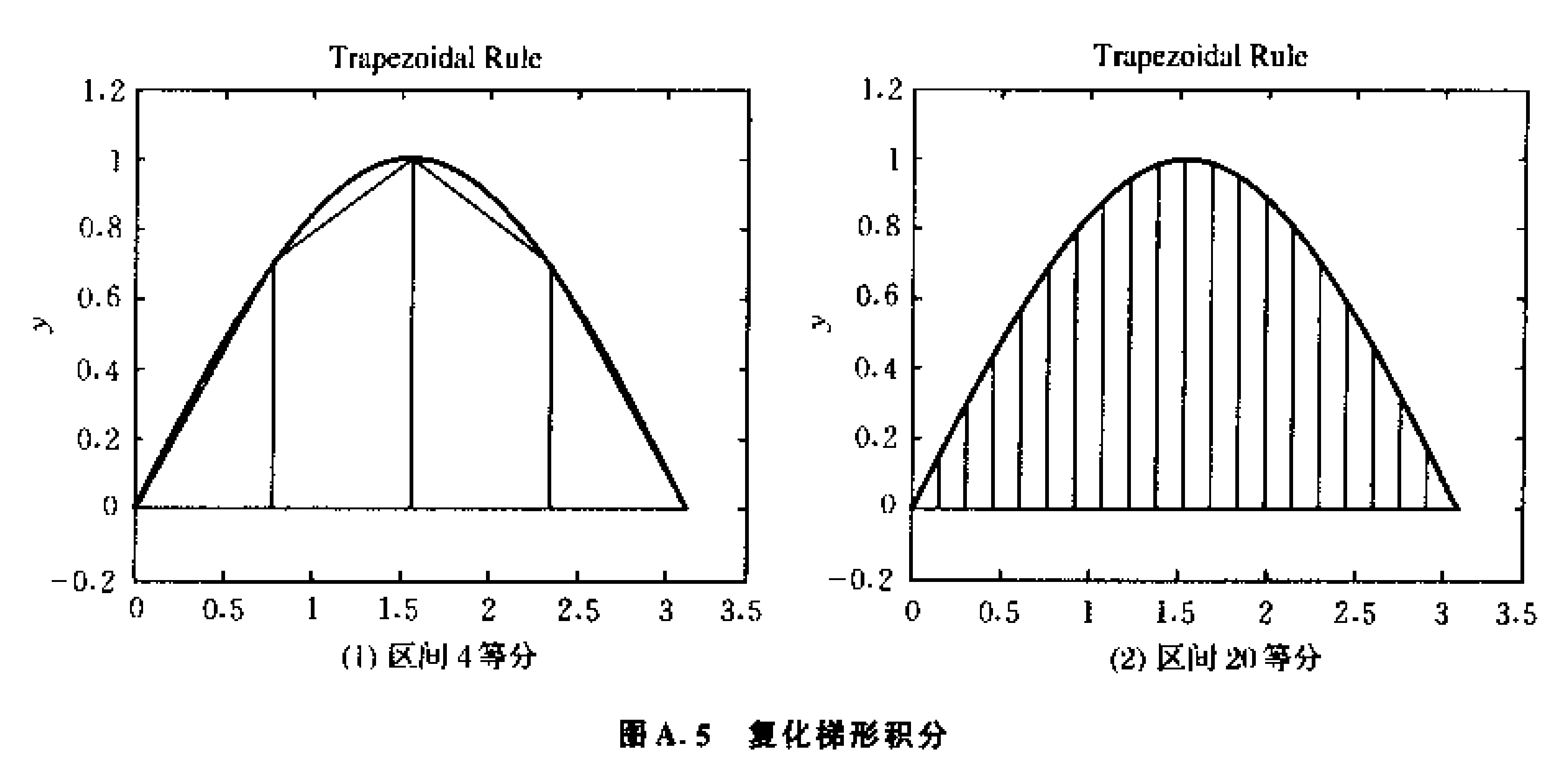
计算结果为：

```
I= 1.9959
```

图A.5表示了复化梯形求积的过程。

2. 复化Simpson积分法程序

程序名称　Simpson.m



(1) 区间4等分　(2) 区间20等分

图A.5　复化梯形积分

调用格式　I=Simpson('f_name',a,b,n)

程序功能　用复化Simpson公式求定积分值。

输入变量　f_name为用户自己编写给定函数y=f(x)的M函数而命名的程序文件名;

　　a为积分下限;

　　b为积分上限;

　　n为积分区间[a,b]划分成小区间的等份数。

输出变量　I为定积分值。

程序:

```
function I=simpson(f_name,a,b,n)
h=(b-a)/n;
x=a+(0:n)*h;
f=feval(f_name,x);
N=length(f)-1;
if N==1
    fprintf('Data has only one interval')
    return;
end
if N==2
    I=h/3*(f(1)+4*f(2)+f(3));
    return;
end
if N==3
    I=3/8*h*(f(1)+3*f(2)+3*f(3)+f(4));
    return;
end
I=0;
if 2*floor(N/2)==N
```

```
    I=h/3*(2*f(N-2)+2*f(N-1)+4*f(N)+f(N+1));
    m=N-3;
else
    m=N;
end
    I=I+(h/3)*(f(1)+4*sum(f(2:2:m))+2*f(m+1));
if m>2
    I=I+(h/3)*2*sum(f(3:2:m));
end
```

例 A.7　求 $I=\int_0^{\pi}\sin x\mathrm{d}x$。

解　先编制 $y=\sin x$ 的 M 函数，程序文件命名为 sin_x.m：

```
function y=sin_x(x)
y=sin(x)
```

将区间 4 等分，调用格式为：

```
I=Simpson('sin_x',0,pi,4)
```

计算结果为：

```
y=
    0 0.7071 1.0000 0.7071 0.0000
I=
    2.0046
```

将区间 20 等分，调用格式为：

```
I=Simpson('sin_x',0,pi,20)
```

计算结果为：

```
y=
        0   0.1564   0.3090   0.4540   0.5878   0.7071   0.8090
   0.8910   0.9511   0.9877   1.0000   0.9877   0.9511   0.8910
   0.8090   0.7071   0.5878   0.4540   0.3090   0.1564   0.0000
I=
    2.0000
```

A.4　常微分方程的数值解法程序

1. 改进的 Euler 法程序

程序名称　Eulerpro.m

调用格式　[X,Y]=Eulerpro('fxy',x0,y0, xend ,h)

程序功能　解常微分方程。

输入变量　fxy 为用户编写给定函数 y=f(x,y)的 M 函数文件名；

x0,xend 为起点和终点；

y0 为已知初始值；

　　　　h 为步长。

输出变量　X 为离散的自变量；

　　　　Y 为离散的函数值。

程序：

```
function[x,y]=Eulerpro(fxy,x0,y0, xend, h)
n=fix((xend-x0)/h);
y(1)=y0;
x(1)=x0;
for k=2:n
    x(k)=0;
    y(k)=0;
end
for i=1:(n-1)
    x(i+1)=x0+i*h;
    y1=y(i)+h*feval(fxy,x(i),y(i));
    y2=y(i)+h*feval(fxy,x(i+1),y1);
    y(i+1)=(y1+y2)/2;
end
plot(x,y)
```

例 A.8　解微分方程 $y'=\sin x$，其中 $x_0=0$，$y_0=-1$。

解　先编制 $y'=\sin x$ 的 M 函数；程序文件命名为 fxy.m：

```
function Z=fxy(x,y)
Z=sin(x);
```

取步长 0.1，调用格式为：

```
[X,Y]=Eulerpro('fxy',0,-1, pi, 0.1)
```

图 A.6　微分方程求解结果

计算结果如图 A.6 所示。

```
x=
     0   0.1000   0.2000   0.3000   0.4000   0.5000   0.6000   0.7000
0.8000   0.9000   1.0000   1.1000   1.2000   1.3000   1.4000   1.5000
1.6000   1.7000   1.8000   1.9000   2.0000   2.1000   2.2000   2.3000
2.4000   2.5000   2.6000   2.7000   2.8000   2.9000   3.0000
y=
-1.0000  -0.9950  -0.9801  -0.9554  -0.9211  -0.8777  -0.8255  -0.7650
-0.6970  -0.6219  -0.5407  -0.4541  -0.3629  -0.2681  -0.1707  -0.0715
 0.0283   0.1279   0.2262   0.3222   0.4150   0.5036   0.5872   0.6649
 0.7359   0.7996   0.8553   0.9025   0.9406   0.9693   0.9883
```

2. Runge-Kutta 法程序

程序名称　RungKt4. m

调用格式　[X,Y]=RungKt4('fxy',x0,y0,xend,M)

程序功能　解常微分方程。

输入变量　fxy 为用户编写给定函数 y'=f(x,y)的 M 函数文件名；
x0,xend 为起点和终点；
y0 为已知初始值；
M 为步长数。

输出变量　X 为离散的自变量；
Y 为离散的函数值。

程序：

```
function [X,Y]=Rungkt4(fxy,x0,y0,xend,M)
h=(xend-x0)/M;
X=zeros(1,M+1);
Y=zeros(1,M+1);
X=x0:h:xend;
Y(1)=y0;
for i=1:M
    k1=h*feval(fxy,X(i),Y(i));
    k2=h*feval(fxy,X(i)+h/2,Y(i)+k1/2);
    k3=h*feval(fxy,X(i)+h/2,Y(i)+k2/2);
    k4=h*feval(fxy,X(i)+h,Y(i)+k3);
    Y(i+1)=Y(i)+(k1+2*k2+2*k3+k4)/6;
end
plot(X,Y)
```

例 A.9　解微分方程 $y'=\sin x$，其中 $x_0=0$，$y_0=-1$。

解　先编制 $y'=\sin x$ 的 M 函数，文件名取为 fxy. m：

```
function Z=fxy(x,y)
Z=sin(x);
```

取步长数为 30，调用格式为：

```
[X,Y]= Rungkt4('fxy',0,-1,pi,30)
```


图 A.7　微分方程的求解过程

计算结果如图 A.7 所示。

```
X=
     0   0.1047   0.2094   0.3142   0.4189   0.5236   0.6283   0.7330
0.8378   0.9425   1.0472   1.1519   1.2566   1.3614   1.4661   1.5708
1.6755   1.7802   1.8850   1.9897   2.0944   2.1991   2.3038   2.4086
2.5133   2.6180   2.7227   2.8274   2.9322   3.0369   3.1416
```

Y=

−1.0000	−0.9945	−0.9781	−0.9511	−0.9135	−0.8660	−0.8090	−0.7431
−0.6691	−0.5878	−0.5000	−0.4067	−0.3090	−0.2079	−0.1045	0.0000
0.1045	0.2079	0.3090	0.4067	0.5000	0.5878	0.6691	0.7431
0.8090	0.8660	0.9135	0.9511	0.9781	0.9945	1.0000	

A.5　线性方程组的数值解法程序

1. 列主元 Guass 消去法

程序名称　Guass.m

调用格式　[U,P,PB,X]=Guass(A,B)

程序功能　解线性方程组 AX=B,其中 A 为非奇异矩阵。

输入变量　A 为 N×N 非奇异矩阵,方程组的系数矩阵;
　　　　　B 为 N×1 矩阵,方程组的常向量。

输出变量　X 为 N x 1 矩阵,方程组的解向量;
　　　　　U 为 N×N 矩阵,A 矩阵消元后形成的上三角矩阵;
　　　　　P 为 N×1 矩阵,选列主元时进行的行变换矩阵;
　　　　　PB 为 N×1 矩阵,选列主元后方程组的常向量。

程序:

```
[N,N]=size(A);
U=zeros(N,N);
X=zeros(N,1);
PB=zeros(N,1);
Swap=zeros(1,N);
Row=1:N;
for r=1:N-1
    [max_a_ir,m]=max(abs(A(r:N,r)));
    Swap=A(r,:);
    A(r,:)=A(r+m-1,:);
    A(r+m-1,:)=Swap;
    U(r,r:N)=A(r,r:N);
    d=Row(r);
    Row(r)=Row(r+m-1);
    Row(r+m-1)=d;
    if A(r,r)==0
        error('A is singular. No unique solution')
        break
    end
    for k=r+1:N
        c=A(k,r)/A(r,r);
```

```
        A(k,r)=c;
        A(k,r+1:N)=A(k,r+1:N)-c*A(r,r+1:N);
    end
    U(N,N)=A(N,N);
end
P=Row';
PB(1)=B(Row(1));
for k=2:N
    PB(k)=B(Row(k))-A(k,1:k-1)*PB(1:k-1);
end
X(N)=PB(N)/A(N,N);
for k=N-1:-1:1
    X(k)=(PB(k)-A(k,k+1:N)*X(k+1:N))/A(k,k);
end
```

例A.10　已知a=[5,-2,1;1,5,-3;2,1,-5],b=[4;2;-11],求aX=b的解。

解

```
>>[U,P,PB,X]=Guass(a,b)
U=
    5.0000   -2.0000    1.0000
         0    5.4000   -3.2000
         0         0   -4.3333
P=
    1
    2
    3
PB =
      4.0000
      1.2000
    -13.0000
X=
    1.0000
    2.0000
    3.0000
```

2. Jacobi 迭代法

程序名称　Jacobi.m

调用格式　[K,X]=Jacobi(A,B,X0,delta,Nmax)

程序功能　解线性方程组AX=B,其中A为非奇异矩阵。

输入变量　A为N×N非奇异矩阵,方程组的系数矩阵;

　　　　　B为N×1矩阵,方程组的常向量;

　　　　　X0为N×1矩阵,迭代初值;

delta 为允许迭代误差;
Nmax 是程序设定的最大迭代次数。

输出变量 X 为N×1 矩阵,方程组的解向量;
K 程序停止运行的实际迭代次数。

程序:

```
function [K,X]= (A,B,X0,delta,Nmax)
N=length(B);
for K=1:Nmax
    for i=1:N
        X(i)=(B(i)-A(i,[1:i-1,i+1:N]) * X0([1:i-1,i+1:N]))/A(i,i);
    end
    err=abs(norm(X'-X0));
    relerr=err/(norm(X)+eps) ;
    X0=X';
    if(err<delta)|(relerr<delta)
        break
    end
end
X=X';
```

例A.11 已知a=[5,-2,1;1,5,-3;2,1,-5],b=[4;2;-11],求aX=b 的解。

解

```
>>[K,X]=Jacobi(a,b,[0;0;0],0.0001,500)
K=
    13
X=
    1.0001
    1.9999
    3.0000
```

3. Guass-Seidel 迭代法

程序名称 GSeidel.m

调用格式 [K,X]=GSeidel(A,B,X0,delta,Nmax)

程序功能 解线性方程组AX=B,其中A 为非奇异矩阵。

输入变量 A 为N×N 非奇异矩阵,方程组的系数矩阵;
B 为N×1 矩阵,方程组的常向量;
X0 为N×1 矩阵,迭代初值;
delta 为允许迭代误差;
Nmax 是程序设定的最大迭代次数。

输出变量 X 为N×1 矩阵,方程组的解向量;
K 程序停止运行的实际迭代次数。

程序:

```
function [K,X]=GSeidel(A,B,X0,delta,Nmax)
N=length(B);
for K=1:Nmax
    for i=1:N
        if i==1
            X(1)=(B(1)-A(1,2:N) * X0(2:N))/A(1,1);
        elseif i==N
            X(N)=(B(N)-A(N,1:N-1) * (X(1:N-1))')/A(N,N);
        else
            X(i)=(B(i)-A(i,1:i-1) * X(1:i-1)'-A(i,i+1:N) * X0(i+1:N))/
                A(i,i);
        end
    end
    err=abs(norm(X'-X0));
    relerr=err/(norm(X)+eps);
    X0=X';
    if (err<delta)|(relerr<delta)
        break
    end
end
X=X';
```

例A.12　已知a=[5,-2,1;1,5,-3;2,1,-5],b=[4;2;-11],求aX=b的解。

解

```
>>[K,X]=GSeidel(a,b,[0;0;0],0.0001,500)
K=
    10
X=
    0.9999
    2.0000
    3.0000
>>
>> [K,X]=GSeidel(a,b,[0;0;0],eps,500)
K=
    34X=
    1
    2
    3
>>
>>[K,X]=Jacobi(a,b,[0;0;0],eps,500)
K=
    49
```

```
X=
      1.0000
      2.0000
      3.0000
```

附录B　部分习题参考答案

习题一

1.1题

(1) $|e|\leqslant 1\times\frac{1}{2}=0.5$, $|e_r|\leqslant\frac{0.5}{4580}=0.1092\times 10^{-2}$, 4位

(2) $|e|\leqslant 0.0001\times\frac{1}{2}=0.00005$, $|e_r|\leqslant\frac{0.00005}{0.0414}=0.00121$, 3位

(3) $|e|\leqslant 0.0001\times 10^3\times\frac{1}{2}=0.05$, $|e_r|\leqslant\frac{0.05}{0.1340\times 10^3}=0.3731\times 10^{-3}$, 4位

或 $|e|\leqslant\frac{1}{2}\times 10^{3-4}=0.05$, $|e_r|\leqslant\frac{0.05}{0.1340\times 10^3}=0.3731\times 10^{-3}$, 4位

(4) $|e|\leqslant 1\times 10^{-2}\times\frac{1}{2}=0.005$, $|e_r|\leqslant\frac{0.005}{2789\times 10^{-2}}=0.1793\times 10^{-3}$, 4位

1.2题　333,3.33,0.00333都有3位有效数字,但0.00333的误差限$\varepsilon_r=0.00001$最小,故其精度最高。

1.3题

(1) 45800和458×10^2不一样,有效数位分别为5位和3位。

(2) 0.00438和0.0438×10^{-1}一样。

(3) 0.4105×10^2和0.04105×10^3一样。

(4) 9800和98×10^2不一样,有效数位分别为4位和2位。

(5) 0.8070和0.807不一样,有效数位分别为4位和3位。

1.4题　$\sqrt{10}\cong 3(1+1/2\times 1/9-1/8\times 1/81)=3.16206$

$e_r=(3.16206-3.1622777)/3.16206=0.00688\%<0.1\%$

1.5题　有效数位为2位。

1.6题　有效数位为3位。

1.7题　有效数位为5位。

1.8题　不能,有效数位被随意增加了。

1.9题　$1/(662*663)=2.278*10^{-6}$

1.10题　$a=(\sqrt{2}-1)^6\cong 0.0050506339$

(1) $99-70\sqrt{2}\cong 99-70\times 1.4=1$

(2) $(3-2\sqrt{2})^3\cong(3-2\times 1.4)^3=0.008$

(3) $1/(3+2\sqrt{2})^3\cong 1/(3+2\times 1.4)^3=0.00512$

(4) $1/(\sqrt{2}+1)^6\cong 1/(1.4+1)^6=0.0052$

(5) $1/(99+70\sqrt{2})\cong 1/(99+70\times 1.4)=0.00508$

习题二

2.1 题　迭代次数n=19，x=1.292496

2.2 题　迭代次数n=10，x=1.325195

2.3 题

(1) 当$0\leqslant x\leqslant 1$时，n=20，x=0.677213，$\varepsilon=9.537e-007$

当$1\leqslant x\leqslant 2$时，n=20，x=1.906804，$\varepsilon=9.537e-007$

(2) 当$-0.5\leqslant x\leqslant 0.5$时，n=20，x=−0.000001，$\varepsilon=9.537e-007$

当$0.5\leqslant x\leqslant 2$时，n=21，x=0.746881，$\varepsilon=7.153e-007$

2.4 题

(1) $|\varphi_1'(x)|=0.5926<1$，收敛。

(2) $|\varphi_2'(x)|=0.4558<1$，收敛。取$x_0=1.5$，$x_1=\varphi_2(x_0)=1.4812$，…，$x_8=\varphi_2(x_7)=1.4656$

(3) $|\varphi_3'(x)|=104142>1$，不收敛。

(4) $|\varphi_4'(x)|=2.1900>1$，不收敛。

2.5 题　$x_{k+1}=x_k-f(x_k)/f'(x_k)$

(1) $x_0=1$，n=4，$x_k=0.67721288996077$

$x_0=1.5$，n=5，$x_k=1.90680389498678$ 或

$x_0=3$，n=5，$x_k=1.90680389497148$

(2) $x_0=-0.5$，n=5，$x_k=-3.840496567839469e-012$

$x_0=1.4$，n=5，$x_k=0.74688174231442$

(3) $x_0=-1$，n=4，$x_k=-0.37141775272400$

$x_0=4.2$，n=5，$x_k=4.70793791814277$

$x_0=1$，n=4，$x_k=0.60526712134827$

(4) $x_0=2$，n=5，$x_k=0.45339765184558$

(5) $x_0=0$，n=3，$x_k=2.00000000000051$

2.6 题

当$y_1=3\sqrt{4-2(x-3)^2}$时，有$f_1(x)=(x-2)^2+(3\sqrt{4-2(x-3)^2}2x-3)^2-5$

当$y_2=-3\sqrt{4-2(x-3)^2}$时，有$f_2(x)=(x-2)^2+(-3\sqrt{4-2(x-3)^2}+2x-3)^2-5$

计算结果如表B.1所示。

表B.1　习题2.6的计算结果

$f(x)=0$	λ	x_0	n	x_k	y
$f_1(x)=0$	0.999	1.5	5	1.65806648	1.89363663
	0.9	1.5	8	1.65806648	1.89363663
$f_2(x)=0$	0.999	1.5	5	1.73622590	−2.69290743
	0.9	1.5	10	1.73622590	−2.69290743
	0.999	3.5	3	3.48288218	−5.63940125
	0.9	3.5	7	3.48288218	−5.63940125
	0.999	4.0	4	4.02873354	−4.11712660
	0.9	4.0	6	4.02873354	−4.11712660

习题三

3.1 题　$p_1(x)=\frac{(x-3)}{1-3}\times 1+\frac{(x-1)}{3-1}\times 2=\frac{x}{2}+\frac{1}{2}$, $p_1(1.5)=1.25$

3.2 题

$$p_2(x)=\frac{(x-3)(x-2)}{2}\times(1)+\frac{(x-1)(x-2)}{2}\times(2)+\frac{(x-1)(x-3)}{-1}\times(-1)$$

$p_2(1.5)=-0.625$

3.3 题　$p_3(x)=0.1000+0.0294-0.0080+0.0000=0.1214$

3.4 题

$$p_2(x)=\frac{(x-1)(x-2)}{(-1)(-2)}\times 1+\frac{(x-0)(x-2)}{1\times(-1)}\times 2+\frac{(x-0)(x-1)}{2\times 1}\times 3=x+1$$

插值结果为一次多项式,3 个数据点在一条直线上。

3.8 题

$$\frac{\partial Q}{\partial x}=\frac{\partial}{\partial x}(\varepsilon_1^2 1+\varepsilon_2^2+\varepsilon_3^2+\varepsilon_4^2)=23x-2y-38=0$$

$$\frac{\partial Q}{\partial y}=\frac{\partial}{\partial y}(\varepsilon_1^2+\varepsilon_2^2+\varepsilon_3^2+\varepsilon_4^2)=-2x+46y-8=0$$

x=1.67362, y=0.24668

习题四

4.1 题

当n = 2, 4, 8, 16 时,计算结果分别是:

(1) 0.3590　0.3498　0.3474　0.3468

(2) 1.7539　1.7272　1.7205　1.7188

(3) 0.4083　0.4062　0.4056　0.4055

4.2 题　0.9481　0.9871　0.9968　0.9997　1.0000

4.9 题　T=0.4268　S=0.4309　C=0.4310　I=0.4310

4.10 题　准确值I=1.7182818

(1) 区间 177 等分　T=1.718286

(2) 区间 14 等分　S=1.718282

4.11 题

(1) $T_1=\frac{3}{2}$, $S_1=1$

(2) $T_2=1.1250$, $S_2=1$

(3) $T_3=1.0556$, $T_4=1.0313$, $T_8=1.0078$

习题五

5.1 题

x=	0	0.1000	0.2000	0.3000
y=	0	0	0.0010	0.0050

5.2 题

(1)	x=	0	0.1000	0.2000	0.3000	0.4000	0.5000
	y=	1.0000	1.1000	1.1890	1.2652	1.3273	1.3742
(2)	x=	0	0.1000	0.2000	0.3000	0.4000	0.5000
	y=	1.0000	0.8000	0.6505	0.5372	0.4501	0.3821
(3)	x=	0	0.1000	0.2000	0.3000	0.4000	0.5000
	y=	0.5000	0.4500	0.4060	0.3694	0.3415	0.3233

5.7 题　$y_1=y_0+\frac{h}{6}(k_1+2k_2+2k_3+k_4)=1+\frac{1}{6}(-1+2\times(-0.6667)+2\times(-0.7059)-0.2707)=0.3307$

5.8 题　将二阶微分方程转化为一阶微分方程组：

$$\begin{cases} y_{n+1}=y_n+hz_n \\ z_{n+1}=z_n+2hy_n^3 \end{cases}\qquad x_0=1, z_0=-1, y_0=-1$$

$x_1=1.1, z_1=-1.2, y_1=-1.1$；　　$y=y(1.1)=-1.11111$

$x_2=1.2, z_2=-1.4662, y_2=-1.22$；　　$y=y(1.2)=-1.25$

$x_3=1.3, z_3=-1.82937, y_3=-1.36662$；　　$y=y(1.3)=-1.42857$

$x_4=1.4, z_4=-2.33984, y_4=-1.54956$；　　$y=y(1.4)=-1.66667$

习题六

6.2 题　不可分解。

6.3 题

(1) X=(1,1,1)，消元过程如下：

$$\begin{pmatrix} 2 & 1 & 1 & 4 \\ 1 & 3 & 2 & 6 \\ 1 & 2 & 2 & 5 \end{pmatrix}\to\begin{pmatrix} 1 & 0.5 & 0.5 & 2 \\ 0 & 2.5 & 1.5 & 4 \\ 0 & 1.5 & 1.5 & 3 \end{pmatrix}\to\begin{pmatrix} 1 & 0.5 & 0.5 & 2 \\ 0 & 1 & 0.6 & 1.6 \\ 0 & 0 & 0.6 & 0.6 \end{pmatrix}\to\begin{pmatrix} 1 & 0.5 & 0.5 & 2 \\ 0 & 1 & 0.6 & 1.6 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

(2) 无解，消元过程如下：

$$\begin{pmatrix} 1 & 1 & -1 & 3 \\ 2 & 1 & -1 & 0 \\ -1 & -2 & 2 & -5 \end{pmatrix}\to\begin{pmatrix} 1 & 1 & -1 & 3 \\ 0 & -1 & 1 & -6 \\ 0 & -1 & 1 & -2 \end{pmatrix}\to\begin{pmatrix} 1 & 1 & -1 & 3 \\ 0 & 1 & -1 & 6 \\ 0 & 0 & 0 & 4 \end{pmatrix}$$

(3) X=(−1.3636,−1.4545,−1.1818)，消元过程如下：

$$\begin{pmatrix} 3 & -1 & 2 & -5 \\ 1 & 1 & 1 & -4 \\ 2 & 1 & -1 & -3 \end{pmatrix}\to\begin{pmatrix} 1 & -\frac{1}{3} & \frac{2}{3} & -\frac{5}{3} \\ 0 & \frac{4}{3} & \frac{1}{3} & -\frac{7}{3} \\ 0 & \frac{5}{3} & -\frac{7}{3} & \frac{1}{3} \end{pmatrix}\to\begin{pmatrix} 1 & -\frac{1}{3} & \frac{2}{3} & -\frac{5}{3} \\ 0 & 1 & \frac{1}{4} & -\frac{7}{4} \\ 0 & \frac{5}{3} & -\frac{7}{3} & -\frac{1}{3} \end{pmatrix}$$

$$\to\begin{pmatrix} 1 & -\frac{1}{3} & \frac{2}{3} & -\frac{5}{3} \\ 0 & 1 & \frac{1}{4} & -\frac{7}{4} \\ 0 & 0 & -33 & 39 \end{pmatrix}$$

6.4 题

(1) X=(3,1,1)，消元过程如下：

$$\begin{bmatrix}2&-1&-1&4\\3&4&-2&11\\3&-2&4&11\end{bmatrix}\to\begin{bmatrix}3&4&-2&11\\2&-1&-1&4\\3&-2&4&11\end{bmatrix}\to\begin{bmatrix}1&\frac{4}{3}&-\frac{2}{3}&\frac{11}{3}\\0&-\frac{11}{3}&\frac{1}{3}&-\frac{10}{3}\\0&-6&6&0\end{bmatrix}$$

$$\to\begin{bmatrix}1&\frac{4}{3}&-\frac{2}{3}&\frac{11}{3}\\0&-6&6&0\\0&-\frac{11}{3}&\frac{1}{3}&-\frac{10}{3}\end{bmatrix}\to\begin{bmatrix}1&\frac{4}{3}&-\frac{2}{3}&\frac{11}{3}\\0&1&-1&0\\0&0&-10&-10\end{bmatrix}\to\begin{bmatrix}1&\frac{4}{3}&-\frac{2}{3}&\frac{11}{3}\\0&1&-1&0\\0&0&1&1\end{bmatrix}$$

6.5 题

(1) n=30, x=(0.9999,2.0000,3.0000)

(2) n=15, x=(1.0000,−1.0000)

(2) n=11, x=(1.5000,2.0000,−4.2000,6.0000)

6.6 题

(1) n=10, x=(0.9999,2.0000,3.0000)

(2) n=6, x=(1.0000,−1.0000)

(3) n=2, x=(1.5000,2.0000,−4.2000,6.0000)

6.9 题　x=(−4,3,2)

6.10 题　(A|E)(E|A^{-1})

6.11 题

(1) rank(a)=3, x=(2.5,4,3.5)

(2) rank(a)=3, x=(−2,2.333,10.333)

参考文献

[1] 崔国华,许如初. 计算方法. 北京:电子工业出版社,2001

[2] 马东升. 数值计算方法. 北京:机械工业出版社,2002

[3] 吴勃英. 数值分析原理. 北京:科学出版社,2003

[4] John H. Mathews,Kurtis D. Fink. 数值方法(MATLAB版)(第三版). 陈渝,周璐,钱方等译. 北京:电子工业出版社,2002

[5] 张德荣,王新民,高安民. 计算方法与算法语言. 北京:高等教育出版社,1981

[6] 何汉林,魏汝祥,李卫军. 数值分析. 武汉:湖北科学技术出版社,1999

[7] 王能超. 数值分析简明教程. 北京:高等教育出版社,1984

[8] 孙志忠,袁慰平,闻震初. 数值分析(第二版). 南京:东南大学出版社,2002

[9] 张诚坚,高健,何南忠. 计算方法. 北京:高等教育出版社,Springer出版社,1999

[10] 金聪,熊盛武. 数值分析. 武汉:武汉理工大学出版社,2003

[11] 封建湖,车刚明,聂玉峰. 数值分析原理. 北京:科学出版社,2001

[12] 史万明,杨骅飞,吴裕树,孙新. 数值分析. 北京:北京理工大学出版社,2002

[13] 王沫然. MATLAB与科学计算(第二版). 北京:电子工业出版社,2003

[14] 龚剑,朱亮. MATLAB 5.X入门与提高. 北京:清华大学出版社,2000

[15] 陈怀琛,吴大正,高西全. MATLAB及在电子信息课程中的应用. 北京:电子工业出版社,2003

[16] 张晓华. 控制系统数字仿真与CAD. 北京:机械工业出版社,2003

[17] 苏金明,阮沈勇. MATLAB 6.1实用指南(上册). 北京:电子工业出版社,2002

［Ｇｅｎｅｒａｌ Ｉｎｆｏｒｍａｔｉｏｎ］
书名＝新世纪高等学校计算机系列教材　数值计算方法　（ＭＡＴＬＡＢ语言版）
作者＝李林
页数＝１９３
ＳＳ号＝１１５８４６２３
出版日期＝２００６年０２月第１版